Volume 19 Number 3

ACTA CYBERNETICA

Szeged, 2010

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. Fifty reprints are supplied for each article published.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage http://www.inf.u-szeged.hu/actacybernetica/ .

Zoltán Ésik
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
ze@inf.u-szeged.hu

Zoltán Fülöp
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
fulop@inf.u-szeged.hu

Ferenc Gécseg
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
gecseg@inf.u-szeged.hu

Jozef Gruska
Institute of Informatics/Mathematics
Slovak Academy of Science
Bratislava, Slovakia
gruska@savba.sk

Tibor Gyimóthy
Department of Software Engineering
University of Szeged
Szeged, Hungary
gyimothy@inf.u-szeged.hu

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
helmut@csd.uwo.ca

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
vogler@inf.tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Symposium of Young Scientists on Intelligent Systems

Guest Editors:

Tibor Gregorics

Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
gt@inf.elte.hu

Péter Szeredi

Department of Computer Science
and Information Theory
Budapest University of Technology
and Economics
Budapest, Hungary
szeredi@cs.bme.hu

Bálint Molnár

Department of Information Systems
Corvinus University of Budapest
Budapest, Hungary
molnar@informatika.uni-corvinus.hu

Zoltán Vámossy

Institute of Software Technology
Óbuda University
Budapest, Hungary
vamossy.zoltan@nik.uni-obuda.hu

Edit Sántáné-Tóth

Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
santa@inf.elte.hu

László Zsolt Varga

System Development Department
Computer and Automation Research
Institute of the Hungarian Academy
of Sciences (MTA SZTAKI)
Budapest, Hungary
laszlo.varga@sztaki.hu

Preface

This issue of *Acta Cybernetica* contains two papers selected from presentations at the Third and Fourth Symposia of Young Scientists on *Intelligent Systems*, held in Budapest, on November 28, 2008 and November 20, 2009, respectively. The Symposia, with the Hungarian acronyms IRFIX'08 and IRFIX'09, were organised by the Artificial Intelligence Section of the John von Neumann Computer Society (JvNCS), the Hungarian member of the European Coordinating Committee for Artificial Intelligence (ECCAI). The Programme Committee was led by the Chair of the AI Section of JvNCS, Péter Szeredi, and included the five members of the Executive Board of the AI Section, listed as Guest Editors below.

The main goal of this series of meetings is to provide a forum for young researchers in both theoretical and practical AI for presenting their work, and to support the exchange of ideas between the Hungarian research groups in AI. The Symposium was part of the Hungarian Science Festival, a month-long series of lectures, conferences, and celebrations.

The IRFIX'08 Symposium included 6 talks and 6 poster presentations, while the IRFIX'09 event featured 7 talks and 9 poster presentations. Both Symposia included invited lectures: in 2008 Barnabás Takács presented a talk on virtual people and their practical applications, while the 2009 lecture by Árpád Bedő and Bálint Dömölki was entitled: "When Everything is Programmable – an account on a conference in California". There were over 50 participants at both Symposia, representing a broad range of Hungarian higher education and research institutions, as well as company research labs. There was a lively discussion after each talk, which continued during the concluding poster session.

The event was held in the Bécsi út building of the John von Neumann Faculty of Informatics of Budapest Tech. (Budapest Tech received the University status on January 1, 2010, and became Óbuda University, `http://www.uni-obuda.hu/en/`.)

The authors of both standard and poster presentations were invited to submit papers to a Special Issue of *Acta Cybernetica*. Six papers were received and were then subjected to the normal refereeing process of the Journal. The two accepted papers deal with the topic of Ontology management – a research area which belongs to the rapidly evolving field of Semantic Technologies. The paper by Zombori discusses a reasoning algorithm for the description logic language $\mathcal{SHQ}$, which maps to an important subset of the OWL Web Ontology Language. The paper by Simonyi and Szőts presents an ontology segmentation tool, which helps in building and integrating ontologies.

Thanks are due to all the authors presenting their work at IRFIX Symposia and especially those submitting papers to this Special Issue. In addition to the Guest Editors, the following colleagues took part in the process of reviewing: Gergely

Lukácsy, Katalin Pásztor Varga, Miklós Szőts, and Zsolt Zombori. Their help is very much appreciated. Special thanks are due to Zoltán Vámossy, for his excellent work on the local organisation of the event.

The Fifth Symposium of Young Scientists, *Intelligent Systems 2010,* will be held on November 26, 2010, and the next Special Issue of Acta Cybernetica is scheduled to appear in mid-2011.

Tibor Gregorics
Bálint Molnár
Edit Sántáné-Tóth

Péter Szeredi
Zoltán Vámossy
László Zsolt Varga

Guest Editors
Members of the Executive Board of the
Artificial Intelligence Section of JvNCS

Acta Cybernetica 19 (2010) 571–588.

A Resolution Based Description Logic Calculus

Zsolt Zombori*

Abstract

We present a resolution based reasoning algorithm called *DL calculus* that decides concept satisfiability for the $\mathcal{SHQ}$ language. Unlike existing resolution based approaches, the DL calculus is defined directly on DL expressions. We argue that working on this high level of abstraction provides an easier to grasp algorithm with less intermediary transformation steps and increased efficiency. We give a proof of the completeness of our algorithm that relies solely on the $\mathcal{ALCHQ}$ tableau method, without requiring any further background knowledge.

1 Introduction and background

The Tableau Method [1] has long provided the theoretical background for DL reasoning and most existing DL reasoners implement some of its numerous variants. The typical DL reasoning tasks can be reduced to consistency checking and this is exactly what the Tableau Method provides. While the Tableau itself has proven to be very efficient, the reduction to consistency check is rather costly for some reasoning tasks. In particular, the ABox reasoning task *instance retrieval* requires running the Tableau Method for every single individual that appears in the knowledge base. Several techniques have been developed to make tableau-based reasoning more efficient on large data sets, (see e.g. [4]), that are used by the state-of-the-art DL reasoners, such as RacerPro [5] or Pellet [11].

Other approaches use first-order resolution for reasoning. A resolution-based inference algorithm is described in [7] which is not as sensitive to the increase of the ABox size as the tableau-based methods. The system KAON2 [10] is an implementation of this approach, providing reasoning services over the description logic language $\mathcal{SHIQ}$. The algorithm used in KAON2 in itself is not any more efficient for instance retrieval than the Tableau, but several steps that involve only the TBox can be performed before accessing the ABox, after which some axioms can be eliminated because they play no further role in the reasoning. This yields a qualitatively simpler set of axioms which then can be used for an efficient, query driven data reasoning. For the second phase of reasoning KAON2 uses a disjunctive datalog engine and not the original calculus. Thanks to the preprocessing, query

*Budapest University of Technology and Economics, Department of Computer Science and Information Theory, E-mail: zombori@cs.bme.hu

answering is very focused, i.e., it accesses as little part of the ABox as possible. However, in order for this to work, KAON2 still needs to go through the whole ABox once at the end of the first phase.

Reading the whole ABox is not a feasible option in case the ABox is bigger than the available memory or the content of the ABox changes so frequently that on-the-fly ABox access is an utmost necessity. Typical such scenarios include reasoning on web-scale or using description logic ontologies directly on top of existing information sources, such as in a DL based information integration system.

We have developed a DL ABox reasoner called DLog [9], available to download at `http://dlog-reasoner.sourceforge.net`, which is built on similar principles to KAON2. We will only highlight two main differences. First, instead of a datalog engine, we use the reasoning mechanism of the Prolog language [3] to perform the second phase (see [8]). Second, we use a modified resolution calculus (see [12]) that allows us to perform more inference steps in the first phase, thanks to which more axioms can be eliminated, yielding an even simpler set of axioms to work with in the second phase. The important difference is that while the approach of [10] can only guarantee that there are no nested functional symbols, our calculus ensures that no function symbols remain at all. This makes the subsequent reasoning easier and we can perform focused, query driven reasoning without any transformation that would require going through the ABox even once.

[12] describes the first phase of the reasoning algorithm implmented in DLog. The DL calculus presented in [13] aims to improve on this algorithm. We move the resolution-based reasoning from the level of first-order clauses to DL axioms, which saves us many intermediary transformation steps. Our current paper is the revised and corrected version of [13]. To avoid a problem in the proof published in [13], we had to restrict the calculus from the $\mathcal{SHIQ}$ language to the $\mathcal{SHQ}$ language. We hope to lift this restriction in the near future.

Our work is yet incomplete in that we only provide an algorithm for TBox reasoning. Although we sketch an extension to ABox reasoning at the end of the paper, we do not yet have a proof for its correctness.

This paper is structured as follows. First, in Section 2 we give a brief introduction to Description Logics and in particular the $\mathcal{SHQ}$ language. In Section 3 we present the DL calculus that performs consistency check for a $\mathcal{SHQ}$ TBox, and show that it can also be used to decide concept satisfiability. In Section 4 we discuss the time complexity of the algorithm. In Section 5 we prove the soundness of the DL calculus. In Section 6 we prove that the calculus is complete. Section 7 introduces our future work, in particular the extension of the DL calculus to ABox reasoning. Finally, Section 8 concludes by giving a brief summary of our results.

2 Description Logic

Description Logics (DLs) is a family of simple logic languages used for knowledge representation (for a detailed introduction see [1]). The language expressions use two main building blocks: *atomic concepts* that represent sets of objects and *atomic*

roles that are used to describe relations between objects and stand for sets of object pairs. These building blocks can be combined to create *composite concepts* – as well as *composite roles* for some DL variants.

2.1 Terminological Axioms and Assertions

A DL statement can be an assertion about concrete individuals or it can express some general knowledge, very much like a rule. Statements of the first kind are called *data assertions* that are altogether referred to as the *Assertion box*, or *ABox*. Rule-like statements are called *terminology axioms* that constitute the *Terminology box*, or *TBox*.

2.2 The $\mathcal{SHQ}$ language

All that has been said so far is true of all DL languages. The members of the language family differ in the constructors that are available for building composite concepts and roles. We will be concerned with the $\mathcal{SHQ}$ language and give a brief summary to its syntax.

Consider a set N_C of atomic concepts, N_R of atomic roles and finitely many constant names. Using nothing but these, we can already make simple assertions. We can describe that an individual satisfies some atomic concept ($A(a)$), some atomic role holds between two individuals ($R(a, b)$), two individuals are equal ($a = b$) or they are distinct ($a \neq b$). We can declare some role to be transitive (Trans(R)). We can force a role to be subsumed by another ($R_1 \sqsubseteq R_2$), i.e., that every object pair that satisfies the first role also satisfies the second.

Let $\sqsubseteq^*$ the reflexive-transitive closure of the $\sqsubseteq$ relation. A role R is said to be *simple* if there is no role S such that Trans(S) and $S \sqsubseteq^* R$ hold.

There are no role constructors in the $\mathcal{SHQ}$ language. However, the following concept constructions are available:

A	Atomic concept, $A \in N_C$
$\top$	top (universal concept)
$\bot$	bottom (empty concept)
$\neg C$	complement of C
$C_1 \sqcap C_2$	intersection of C_1 and C_2
$C_1 \sqcup C_2$	union of C_1 and C_2
$\forall R.C$	value restriction (R, C arbitrary role and concept)
$\exists R.C$	existential resctriction (R, C arbitrary role and concept)
$\leq nS.C$	at-most number restriction (S simple role, C arbitrary concept)
$\geq nS.C$	at-least number restriction (S simple role, C arbitrary concept)

Two arbitrary concepts (simple or composite) can be asserted to be equivalent ($C \equiv D$), or that the one is subsumed by the other ($C \sqsubseteq D$). In summary, we give the valid statements of the $\mathcal{SHQ}$ language in Table 1.

Table 1: $\mathcal{SHQ}$ axioms

$\mathcal{SHQ}$ TBox	
$C \sqsubseteq D$	C, D arbitrary concepts
$C \equiv D$	C, D arbitrary concepts
$R \sqsubseteq S$	R, S arbitrary roles
Trans(R)	R arbitrary role
$\mathcal{SHQ}$ ABox	
$C(a)$	C arbitrary concept
$R(a,b)$	R arbitrary role
$a = b$	
$a \neq b$	

3 The DL Calculus

In this section we present a reasoning algorithm, called *DL calculus*, which decides the consistency of a $\mathcal{SHQ}$ TBox. Evidently, such an algorithm can be used for deciding concept satisfiability as well. To see this, suppose we want to know whether concept C is satisfiable in the presence of TBox $\mathcal{T}$. Take a role R that appears neither in $\mathcal{T}$ nor in C. Let us consider a new TBox $\mathcal{T}' = \mathcal{T} \cup \{\top \sqsubseteq \exists R.C\}$. Given that R is a new role name, it is easy to see that the newly added axiom will only introduce inconsistency to the TBox if C is unsatisfiable. C is satisfiable in the presence of TBox $\mathcal{T}$ if and only if $\mathcal{T}'$ is consistent. Hence, by giving an algorithm for TBox consistency check, we also provide an algorithm for concept satisfiability check.

The algorithm can be summarized as follows. We determine a set of concepts that have to be satisfied by each individual of an interpretation in order for the TBox to be true. Next, we introduce inference rules that derive a new concept from two concepts. Using the inference rules, we saturate the knowledge base, i.e., we apply the rules as long as possible and add the consequent to the knowledge base. We also apply redundancy elimination: whenever a concept extends another, it can be safely eliminated from the knowledge base [2]. It can be shown that saturation terminates. We claim that the knowledge base is inconsistent if and only if the saturated set contains the empty concept ($\bot$).

3.1 Preprocessing

We first eliminate transitivity from the knowledge base. It can be shown (see [10]) that any $\mathcal{SHQ}$ knowledge base KB can be transformed into a knowledge base KB' that contains no transitivity axioms and KB' is satisfiable if and only if KB is satisfiable.

Next, we internalize the TBox, i.e., we transform all axioms into a set of concepts that have to be satisfied by each individual. For instance, the axiom $C \sqsubseteq D$ is

equivalent to the axiom $\top \sqsubseteq \neg C \sqcup D$, which amounts to saying that $\neg C \sqcup D$ has to be satisfied by all individuals.

Internalization is followed by structural transformation which eliminates the nesting of composite concepts into each other. A $\mathcal{SHQ}$ expression that appears in the TBox can be of arbitrary complexity, i.e., all sorts of composite concepts can appear within another concept. This makes reasoning very difficult. To solve this problem, we eliminate nesting composite concepts into each other by introducing new concept symbols that serve as names for embedded concepts. For details, see [10].

Finally, we make a small syntactic transformation: concepts $\forall R.C$ and $\exists R.D$ are replaced with equivalent concepts $(\leq 0R.\neg C)$ and $(\geq 1R.D)$, respectively. As a result, we obtain the following types of concepts, where L is a possibly negated atomic concept and R an arbitrary role:

$$L_1 \sqcup L_2 \sqcup \cdots \sqcup L_i$$
$$L_1 \sqcup (\geq kR.L_2)$$
$$L_1 \sqcup (\leq nR.L_2)$$

3.2 Notation

Before presenting the inference rules, we define some important notions. A *literal concept* (typically denoted with L) is a possibly negated atomic concept. A *bool concept* contains no role expressions (allowing only negation, union and intersection). We use capital letters from the beginning of the alphabet $(A, B, C \ldots)$ to refer to bool concepts. In the following, we will always assume that a bool concept is presented in a simplest disjunctive normal form, i.e., it is the disjunction of conjunctions of literal concepts. So for example, instead of $A \sqcup A \sqcup (B \sqcap \neg B \sqcap C)$ we write A, and $A \sqcap \neg A$ is replaced with $\bot$. To achieve this, we apply eagerly the simplification rules presented in Figure 2 (see Subsection 3.5). When the inference rules (see Figure 1) do not preserve disjunctive normal form (DNF), we will use the explicit *dnf* operator:

$$dnf(A \sqcap B) = \begin{cases} dnf(A_1 \sqcap B) \sqcup dnf(A_2 \sqcap B) & \text{if } A = A_1 \sqcup A_2 \\ dnf(A \sqcap B_1) \sqcup dnf(A \sqcap B_2) & \text{if } B = B_1 \sqcup B_2 \\ (A \sqcap B) & \text{otherwise} \end{cases}$$

The *dnf* operator is defined only for concepts that are the intersection of two concepts. The bool concepts in the premises are always in DNF and the conclusion contains either the union or the intersection of such concepts. The union of two DNF concepts is also in DNF so we only need to apply the *dnf* operator to transform the intersection of two DNF concepts.

3.3 Ordering

Let $\succ$ be a total ordering, called a *precedence*, on the set of (atomic concept, atomic role, natural number, logic) symbols, such that $\geq \succ \leq \succ R \succ n \succ C \succ \neg \succ \sqcup \succ$

$\sqcap \succ \top \succ \bot$ for any atomic concept C, atomic role name R and natural number n; furthermore for any two natural numbers $n_1 \succ n_2$ if and only if $n_1 > n_2$. We define a corresponding *lexicographic path ordering* $\succ_{lpo}$ (see [2]) as follows:

$s = f(s_1, \ldots, s_m) \succ_{lpo} g(t_1, \ldots, t_n) = t$ if and only if

1. $f \succ g$ and $s \succ_{lpo} t_i$, for all i with $1 \leq i \leq n$; or
2. $f = g$ and, for some j, we have $(s_1, \ldots, s_{j-1}) = (t_1, \ldots, t_{j-1}), s_j \succ_{lpo} t_j$, and $s \succ_{lpo} t_k$, for all k with $j < k \leq n$; or
3. $s_j \succeq_{lpo} t$, for some j with $1 \leq j \leq m$.

In order for the above definition to be applicable, we treat concept $(\geq kS.A)$ as $\geq(k, S, A)$ and concept $(\leq nR.D)$ as $\leq(n, R, D)$. If the precedence is total on the symbols of the language, then the lexicographic path ordering is total on DL expressions. For simplicity, we often write $\succ$ instead of $\succ_{lpo}$ when it does not lead to confusion. Note a couple properties of our ordering that will be useful later:

1. A $\geq$-concept is greater than any $\leq$-concept or any bool concept.
2. A $\leq$-concept is greater than any bool concept.
3. $C_1 = (\leq n_1 R_1.A_1)$ is greater than $C_2 = (\leq n_2 R_2.A_2)$ if and only if:
   - $R_1 \succ R_2$ or
   - $R_1 = R_2$ and $n_1 > n_2$ or
   - $R_1 = R_2$, $n_1 = n_2$ and $A_1 \succ A_2$

Definition 1 (maximal concept). *Given a set N of concepts, concept $C \in N$ is* maximal *in N if C is greater than any other concept in N.*

Since the ordering $\succ_{lpo}$ is total, for any finite set N there is always a unique concept $C \in N$ that is maximal in N.

3.4 $\mathcal{SHQ}$-concepts

A derivation in the DL calculus generates concepts that are more general than the ones obtained after preprocessing (see Subsection 3.1). We call this broader set *$\mathcal{SHQ}$-concepts*, defined as follows (C, D, E stand for concepts containing no role expressions):

$$C \qquad \text{(bool concepts)}$$

$$C \sqcup \bigsqcup(\leq nR.D) \qquad (\leq\text{-max concepts})$$

$$C \sqcup \left(\bigsqcup(\leq nR.D)\right) \sqcup (\geq kS.E) \qquad (\geq\text{-max concepts})$$

where bool concepts C, D, E are in DNF. Note two important properties of $\mathcal{SHQ}$-concepts:

1. A $\mathcal{SHQ}$-concept is a disjunction that contains at most one $\geq$-concept.

2. There are no nested concepts containing role expressions, i.e., a concept embedded into a $\geq$-concept or a $\leq$-concept is always a bool concept.

According to the ordering defined in Subsection 3.3, each $\leq$-concept is greater than any bool concept, so the maximal disjunct in a $\leq$-max concept is a $\leq$-concept. Similarly, any $\geq$-concept is greater than any $\leq$- or bool concept, so the maximal disjunct in a $\geq$-max concept is a $\geq$-concept. This is the rationale for naming these concepts $\leq$-max and $\geq$-max, respectively.

Obviously, any concept obtained after preprocessing is a $\mathcal{SHQ}$-concept:

Proposition 1. *For any $\mathcal{SHQ}$ knowledge base KB, if we apply the transformations described in Subsection 3.1 on KB, we obtain a set of $\mathcal{SHQ}$-concepts.*

3.5 Inference Rules

The inference rules are presented in Figure 1, where C_i, D_i, E_i are possibly empty bool concepts. W_i stands for an arbitrary $\mathcal{SHQ}$-concept that can be empty as well. Some of the rules do not preserve the disjunctive normal form (DNF) of bool concepts. In such cases, we use the *dnf* operator as defined in Subsection 3.2. Note that two disjunctive concepts are resolved along their respective maximal disjuncts and the ordering that we imposed on the concepts yields a selection function. Since the odering is total, we can always select the unique maximal disjunct to perform the inference step.

Along with the inference rules, we use a futher set of rules that we call *simplification rules* and which are shown in Figure 2. These rules only have one premise which is redundant in the presence of the conclusion and hence can be eliminated. In other words, the simplification rules are used to simplify concepts and do not deduce new concepts. Simplification rules are applied not only to $\mathcal{SHQ}$-concepts, but also to subconcepts appearing in $\mathcal{SHQ}$-concepts. For example, S1 is used to replace the concept $C \sqcup A \sqcup A$ with $C \sqcup A$, but also to replace $(\geq nR.(C \sqcup A \sqcup A))$ with $(\geq nR.(C \sqcup A))$.

Rule1 corresponds to the classical resolution inference and Rule2 makes this same inference possible for entities whose existence is required by $\geq$-concepts. Rule3 and Rule4 are harder to understand. They address the interaction between $\geq$-concepts and $\leq$-concepts. Intuitively, if some entity satisfies $\leq nR.C$ and also satisfies $\geq kS.D$, then there is a potential for clash if concepts C and D are related, more precisely if D is subsumed by C. In such cases $D \sqcap \neg C$ is not satisfiable, which either leads to contradiction if $n < k$ (Rule3) or results in a tighter cardinality restriction on the entity (Rule4). If several $\geq$-concepts and a $\leq$-concept are inconsistent together, then each $\geq$-concept is used to deduce a $\leq$-concept with smaller cardinality (Rule4) until the $\leq$-concept completely disappears from the conclusion (Rule3) and we obtain the empty concept.

Rule1 $\dfrac{C_1 \sqcup (D_1 \sqcap A) \qquad C_2 \sqcup (D_2 \sqcap \neg A)}{C_1 \sqcup C_2}$

where $D_1 \sqcap A$ is maximal in $C_1 \sqcup (D_1 \sqcap A)$
and $D_2 \sqcap \neg A$ is maximal in $C_2 \sqcup (D_2 \sqcap \neg A)$

Rule2 $\dfrac{C \qquad W \sqcup (\geq nR.D)}{W \sqcup (\geq nR.dnf(D \sqcap E))}$

where E is obtained by using Rule1 on premises C and D

Rule3 $\dfrac{W_1 \sqcup (\leq nR.C) \qquad W_2 \sqcup (\geq kS.D)}{W_1 \sqcup W_2 \sqcup (\geq (k-n)S.dnf(D \sqcap \neg C))}$

$n < k, S \sqsubseteq^* R$, $(\leq nR.C)$ is maximal in $W_1 \sqcup (\leq nR.C)$
and $(\geq kS.D)$ is maximal in $W_2 \sqcup (\geq kS.D)$

Rule4 $\dfrac{W_1 \sqcup (\leq nR.C) \qquad W_2 \sqcup (\geq kS.D)}{W_1 \sqcup W_2 \sqcup (\leq (n-k)R.dnf(C \sqcap \neg D)) \sqcup (\geq 1S.dnf(D \sqcap \neg C))}$

$n \geq k, S \sqsubseteq^* R$, $(\leq nR.C)$ is maximal in $W_1 \sqcup (\leq nR.C)$
and $(\geq kS.D)$ is maximal in $W_2 \sqcup (\geq kS.D)$

Figure 1: TBox inference rules of the DL calculus

S1 $\dfrac{C \sqcup L \sqcup \cdots \sqcup L}{C \sqcup L}$

S2 $\dfrac{C \sqcup D \sqcup (D \sqcap E)}{C \sqcup D}$

S3 $\dfrac{C \sqcup D \sqcup (\neg D \sqcap E)}{C \sqcup D \sqcup E}$

S4 $\dfrac{C \sqcup D \sqcup \neg D}{\top}$

S5 $\dfrac{C \sqcup (D \sqcap E \sqcap \neg E)}{C}$

S6 $\dfrac{W \sqcup (\geq nR.\bot)}{W}$

S7 $\dfrac{W \sqcup (\leq nR.\bot)}{\top}$

Figure 2: TBox simplification rules of the DL calculus

3.6 Saturation

We saturate the knowledge base, i.e., we apply the rules in Figure 1 to deduce new concepts as long as possible. Before adding the consequent to the concept set, we eagerly apply the simplification rules of Figure 2 to make the concept as simple as possible. We claim that the consequent is always a $\mathcal{SHQ}$-concept.

Proposition 2. *The set of $\mathcal{SHQ}$-concepts is closed under the inference rules in Figure 1 and the simplification rules in Figure 2.*

Proof. Consider Rule1. $D_1 \sqcap A$ is maximal in $C_1 \sqcup (D_1 \sqcap A)$ which is only possible if C_1 does not contain any $\geq$- or $\leq$-concepts. Hence it is a bool concept. Analogously, the fact that $D_2 \sqcap \neg A$ is maximal in $C_2 \sqcup (D_2 \sqcap \neg A)$ ensures that C_2 is another bool concept. Bool concepts are in DNF. The conclusion is the disjunction of two bool concepts $(C_1 \sqcup C_2)$ which is also in DNF and hence is a bool concept.

Rule2 resolves a bool concept with a $\geq$-max concept. We have just seen that resolving C and D by Rule1 yields a bool concept. We take the conjunction of this concept and another bool concept $(D \sqcap E)$ which is not in DNF, but it yields a bool concept once we apply the *dnf* operator. Hence the conclusion is a $\geq$-max concept.

In Rule3, the maximal disjunct of the first premise is $(\leq nR.C)$, so it does not contain any $\geq$-concept. The second premise is a $\geq$-max concept and contains exactly one $\geq$-concept, namely $(\geq kS.D)$. The conclusion contains one $\geq$-concept and is a $\geq$-max concept. Again, the *dnf* operator is used to ensure that the bool concept appearing in the $\geq$-disjunct of the conclusion is in DNF.

In Rule4, the maximal disjunct of the first premise is $(\leq nR.C)$, so it is a $\leq$-max concept and does not contain any $\geq$-concept. The second premise contains exactly one $\geq$-concept, so W_2 contains no $\geq$-concept. Consequently, the conclusion will contain only one $\geq$-concept and all subconcepts inside $\geq$- and $\leq$-concepts are bool concepts. We obtain a $\geq$-max concept.

Simplification rules S1-S5 eliminate some disjuncts or conjuncts from bool concepts in DNF. The conclusion is always a simpler bool concept in DNF. S6 eliminates an unsatisfiable branch from a disjunction, turning a $\geq$-max concept either to a bool concept or to a $\leq$-max concept. In case of S7, the premise is a tautology and can be safely eliminated. □

4 Termination

The following proposition – along with Propostion 2 – ensures that the DL calculus terminates.

Proposition 3. *The set of all $\mathcal{SHQ}$-concepts that can be deduced from any finite TBox is finite.*

Proof. For any finite TBox, there can only be finitely many distinct role expressions and bool concepts. Furthermore, note that each inference rule either leaves the arity of a number restriction unaltered or reduces it. So in a $(\leq nR.C)$ or $(\geq nR.C)$

expression the number of possible values for n, R and C is finite for a fixed TBox. As all $\mathcal{SHQ}$-concepts are disjunctions of bool, $\leq$, and $\geq$-concepts, we have an upper limit for the set of deducible $\mathcal{SHQ}$-concepts. □

DL calculus deduces only $\mathcal{SHQ}$ concepts from $\mathcal{SHQ}$ concepts. Since there are finitely many $\mathcal{SHQ}$ concepts, even if we have to deduce every possible $\mathcal{SHQ}$-concept, it still requires finitely many steps, so the calculus is guaranteed to terminate.

5 Soundness

It is straightforward to show that the simplification rules are sound, i.e., if all individuals of an interpretation satisfy the premise then they also satisfy the conclusion. We leave this to the reader. The inference rules are slightly more complex.

Theorem 1. *The inference rules of the DL calculus are sound.*

Proof. Consider Rule1 and suppose that x satisfies both premises. Either A or $\neg A$ is true of x. If $A(x)$ is true, then x must satisfy C_2, due to the second premise. Analogously, if $\neg A(x)$ is true, then x must satisfy C_1. In either case, the conclusion holds for x.

We turn to Rule2. Let x be an individual. It satisfies the second premise, so either W or $(\geq nR.D)$ holds for x. In the first case the conclusion is satisfied by x, in the second case x has at least n R-successors that satisfy D. These successors also satisfy the first premise (C) and – given that Rule1 is sound – they satisfy E. If these R-successors satisfy both D and E, then they satisfy $D \sqcap E$ as well. So it holds for x that it has at least n R-successors that satisfy $D \sqcap E$, so the conclusion is again satisfied.

For Rule3, let x be an arbitrary indidivual. If x satisfies either W_1 or W_2, then it satisfies the conclusion. Otherwise, x satisfies $(\leq nR.C)$ and $(\geq kS.D)$, where $S \sqsubseteq R$. So, x has at least k distinct S-successors that satisfy D (that are R-successors as well). Of these, at most n successors can satisfy C, so there are at least $k - n$ S-successors that satisfy $\neg C$. From this it follows directly that the conclusion holds for x.

Finally, let us consider Rule4 and let again x denote an arbitrary individual. If x satisfies either W_1 or W_2, then it satisfies the conclusion. Otherwise, x satisfies $(\leq nR.C)$ and $(\geq kS.D)$, where $S \sqsubseteq R$. So, x has at least k distinct S-successors that satisfy D. If any of these successors satisfy $\neg C$ then the last disjunct of the conclusion holds. Otherwise, all the k S-successors satisfy C. Given that x can have no more than n successors that satisfy C, there cannot be more than $n - k$ successors that are not among those satisfying D, but they satisfy C. Hence the second to last disjunct of the conclusion holds for x. □

6 The Completeness of the DL Calculus

In this section we prove that the method presented in Section 3 is complete, i.e., whenever there is some inconsistency in a TBox $\mathcal{T}$, the empty concept is deduced. We prove completeness by showing that if a saturated set $Sat_{\mathcal{T}}$ does not contain $\bot$ then the axiom $\top \sqsubseteq \bigsqcap Sat_{\mathcal{T}}$ has a model. Instead of building the model itself, we will prove that the $\mathcal{ALCHQ}$ tableau method can find one such model. In order for the model to satisfy $\top \sqsubseteq \bigsqcap Sat_{\mathcal{T}}$, the concepts in $Sat_{\mathcal{T}}$ are added to the label of every newly created node in the tableau.

Although the tableau rules are fairly standard, there might be small variations. Hence, to avoid confusion, in Appendix 8 we provide the definition of the tableau rules that we assume in the following.

6.1 Building the Tableau Tree

In the previous sections, we replaced $\forall$- and $\exists$-concepts with $\leq$- and $\geq$-concepts to make the presentation of the inference rules simpler. As we turn to the tableau, however, the reader might be more familiar with the corresponding $\forall$-rule and $\exists$-rule. Hence, in the following, we will treat our $(\leq 0R.C)$ and $(\geq 1S.D)$ concepts as $(\forall R.\neg C)$ and $(\exists S.D)$, respectively.

Whenever we have several applicable tableau rules, we require the following ordering precedence: $\sqcup$-rules, $\sqcap$-rule, $\exists$-rule, $\geq$-rule, $\forall$-rule, $\bowtie$-rule and $\leq$-rule. When applying the $\sqcup$-rule we proceed with the branch[1] that adds the minimal possible concept to the label of a node. Given that the tableau method is don't care non-deterministic with respect to these choices, the completeness of the algorithm is preserved.

Whenever a node n contains a disjunctive concept $W \sqcup C$, the branch where C is added to the label of n is only examined after each disjunct in W that is smaller than C has been proven unsatisfiable. A *clash* occurs in the tableau tree when an atomic concept name and its negation both appear in the label of some node. In this case we roll back and proceed with another branch. A *final clash* occurs when there are no branches left, i.e., the tableau proves the inconsistency of $Sat_{\mathcal{T}}$. We show that no final clash can be reached if $Sat_{\mathcal{T}}$ does not contain $\bot$.

6.2 Bool Concepts

In the following theorem we consider the case when $Sat_{\mathcal{T}}$ contains only bool concepts.

Theorem 2. *If $Sat_{\mathcal{T}}$ contains only bool concepts and does not contain $\bot$, then no final clash is possible.*

Proof. To obtain contradiction, suppose that we reach a final clash. Hence, for some atomic concept A, both A and $\neg A$ appear in the label of some node. This is

[1] Throughout this paper, "branch" refers to a branch of the meta-tableau tree, i.e., one of the tableaux resulting from the application of a non-deterministic rule.

only possible if $Sat_{\mathcal{T}}$ contains concepts

$$W_1 = C_1 \sqcup (D_1 \sqcap A) \qquad W_2 = C_2 \sqcup (D_2 \sqcap \neg A)$$

The clash is final, so there are no more branches, i.e., $(D_1 \sqcap A)$ and $(D_2 \sqcap \neg A)$ are maximal in W_1 and W_2, respectively, and each disjunct in C_1 and C_2 leads to clash. W_1 and W_2 are resolvable using Rule1, so $Sat_{\mathcal{T}}$ also contains

$$W = C_1 \sqcup C_2$$

W cannot be empty because we assumed that $Sat_{\mathcal{T}}$ does not contain $\bot$. The simplification rules, and in particular S1 was eagerly applied on W_1 and W_2, so there are no other occurrences of $(D_1 \sqcap A)$ in C_1 and $(D_2 \sqcap \neg A)$ in C_2. So the maximal disjuncts in W_1 and W_2 are strictly maximal. Let X denote the greater concept of $(D_1 \sqcap A)$ and $(D_2 \sqcap \neg A)$. X is greater than any disjunct in either C_1 or C_2. This means that the branches corresponding to all disjuncts of W were examined before examining the branch corresponding to X (due to the ordering imposed on the application of the $\sqcup$-rule described in Subsection 6.1). But we know that all disjuncts in W lead to clash, so a final clash must have been obtained on W, even before introducing X to the label of the node, which contradicts our assumption that the final clash involved X. □

Corollary 1. *If $Sat_{\mathcal{T}}$ does not contain $\bot$, then the set of bool concepts in $\mathcal{T}$ is satisfiable.*

Notice that only Rule1 is used to detect the inconsistency of bool concepts. This observation will be useful for us later.

Corollary 2. *If a set N of bool concepts is unsatisfiable then there is a sequence of bool concepts $p_1, p_2 \dots p_n = \bot$ such that for each p_i, there is an instance of Rule1 with premises from $N \cup \{p_1, p_2 \dots p_{i-1}\}$ whose conclusion is p_i. We call this sequence a deduction of $\bot$.*

6.3 $\geq$-max Concepts

Let us now assume that $Sat_{\mathcal{T}}$ contains only bool concepts and $\geq$-max concepts.

Proposition 4. *Let $W = C \sqcup (\geq nR.D)$ be a $\geq$-max concept in $Sat_{\mathcal{T}}$. Then D is satisfiable.*

Proof. Suppose that D is unsatisfiable. Since it is in DNF, it is the disjunction of conjunctions such that each conjunction contains some atom together with its negation. However, the simplification rules are eagerly appled on all $\mathcal{SHQ}$-concepts and due to S5 all disjuncts of D were eliminated. Hence $D = \bot$ and $W = C \sqcup (\geq nR.\bot)$. S6 is applicable on W yielding C, so W was removed from $Sat_{\mathcal{T}}$ and replaced by C. This is a contradiction, so D must be satisfiable. □

Proposition 5. *Let $W = C \sqcup (\geq nR.D)$ be a $\geq$-max concept and $B = \{B_i\}$ a set of bool concepts. If $\{D\} \cup B$ is inconsistent, then there is a deduction of C using Rule1 and Rule2 and the simplification rules.*

Proof. We know from Corollary 2 that there is a deduction $p_1, p_2 \ldots p_n = \bot$ from $\{D\} \cup B$ using Rule1. In this sequence each concept has a set of premises, either from the original concept set or from concepts that were deduced earlier. Let us define the *ancestor* relation as the transitive closure of the premise relation and let *descendant* be its inverse relation. For each p_i, let A_i denote the set of its ancestors that are either identical to D or are descendants of D. For each p_i such that A_i is not the empty set, replace p_i with $C \sqcup (\geq nR.(p_i \sqcap \bigsqcap A_i))$. We obtain a deduction in which each time the conclusion is a $\geq$-max concept, Rule2 is used instead of Rule1. In particular, $p_n = \bot$ is replaced with $C \sqcup (\geq nR.(\bot \sqcap \bigsqcap A_n))$, where the $\geq$-concept is unsatisfiable, so we can deduce C from this concept using the simplification rules (see Proposition 4). □

Corollary 3. *Let $W = C \sqcup (\geq nR.D)$ be a $\geq$-max concept in $Sat_\mathcal{T}$ and let $B = \{B_i\}$ be the set of bool concepts in $Sat_\mathcal{T}$. Then $\{D\} \cup B$ is consistent.*

Proof. Suppose $\{D\} \cup B$ is inconsistent. Then, from Proposition 5, $Sat_\mathcal{T}$ contains C. However, C makes W redundant, so W was eliminated from $Sat_\mathcal{T}$ when C was added to it. This contradicts our assumption that $W \in Sat_\mathcal{T}$. □

Theorem 3. *If $Sat_\mathcal{T}$ contains only bool concepts and $\geq$-max concepts and does not contain $\bot$, then it is consistent.*

Proof. We know from Corollary 1 that the bool concepts are satisfiable. As of the $\geq$-max concepts, at least one of their disjuncts, namely the $\geq$-disjunct can be satisfied: we can create separate successors for each $\geq$-concept, independent of each other (without $\leq$-concepts, these successors never need to be identified). The label of each successor is satisfiable (see Proposition 4 and Proposition 3), so the $\geq$-concept in the parent is satisfiable as well. □

6.4 $\leq$-max Concepts

We now consider a fully general saturated set $Sat_\mathcal{T}$, that might contain bool concepts, $\geq$-max concepts and $\leq$-max concepts. When we build the tableau tree, if a $\leq$-concept appears in the label of a node, we possibly have to add a new concept to the label of a node ($\forall$-rule) or identify two nodes ($\leq$-rule). We show that none of these rules will lead to final clash.

Each successor node is created with an intial concept in its label: for instance, if a new node is created due to concept $\geq 1R.A$, then we call A the *creator concept* of the node. Whatever other concept appears in its label (before performing any identification step), it follows from $A \sqcap \bigsqcap B_i$, where $\{B_i\}$ is the set of bool concepts. If a node with creator concept A has to be identified with another such that the second node contains A in its label, then identification cannot introduce new inconsistency and it can be seen as simply deleting the first node.

As previously, we are only interested in potential clashes that are final. This means that the (non-disjunctive) concepts that are involved in the clash can be assumed to be the maximal disjuncts of $\mathcal{SHQ}$-concepts from $Sat_\mathcal{T}$.

Proposition 6. *Let $Sat_\mathcal{T}$ be a saturated set of $\mathcal{SHQ}$-concepts that does not contain the empty concept $\bot$. Let us try to build a model for $\top \sqsubseteq \bigsqcap Sat_\mathcal{T}$ using the tableau method, observing the restrictions on the order of rules presented in Subsection 6.1. Then we never obtain a final clash.*

Proof. We know from Theorem 3 that the set of bool concepts and $\geq$-max concepts is consistent. Hence, a final clash must involve a $(\leq nR.D)$ concept. We use induction on n, the arity of the $\leq$-concept to show that no final clash is possible.

The base case is when $n = 0$, that is, when we have a $\forall$-concept in the label of a node. The $\forall$-rule fires and a new concept is added to the label of some successors. To obtain contradiction, we assume that this leads to a final clash. Given a node x that has an S-successor y with creator concept A. This means that the label of x contains a concept $\geq kS.A$. Furthermore, the label of x also contains a $\forall$-concept, which is a $(\leq 0R.D)$ concept in our teminology. $S \sqsubseteq R$, so the $\forall$-rule is applicable and puts $\neg D$ in the label of y. We assumed that a clash is obtained, so $A \sqcap \neg D$ is not satisfiable. The $\geq$-concept and $\leq$-concept in the label of x originate from a $\geq$-max and a $\leq$-max concept, respectively, in $Sat_\mathcal{T}$, that is, $Sat_\mathcal{T}$ contains concepts

$$W = E \sqcup (\leq 0R.D) \qquad V = F \sqcup (\geq kS.A)$$

where $(\leq 0R.D)$ is maximal in W, $(\geq kS.A)$ is maximal in V and each disjunct in E and F leads to clash. W and V are resolvable using Rule3 and the conclusion is

$$E \sqcup F \sqcup (\geq kS.dnf(A \sqcap \neg D))$$

$A \sqcap \neg D$ is not satisfiable, so the DL calculus deduces $E \sqcup F$ as well (Proposition 5). However, we know that all disjuncts in E and F lead to clash, so we obtain a final clash without the $\leq$-concept in W. Contradiction.

We now turn to the inductive step. The inductive hypothesis is that a $\leq$-concept can never lead to final clash, i.e., a $(\leq n'R.D)$ concept in the label of a node that is derived from the maximal disjunct of a $\leq$-max concept of$Sat_\mathcal{T}$ can be satisfied for all $n' < n$. We show that this also holds for n.

Let some node x in the tableau tree contain concepts $(\leq nR.D)$ and $(\geq n_i S_i.A_i)$, where $1 \leq i \leq l$ and $S_i \sqsubseteq^* R$. Due to the $(\geq n_i S_i.A_i)$ concepts, we have already created $\Sigma_{i=1}^{l} n_i$ successors with creator concepts $A_1 \ldots A_l$, respectively. D appears in the label of each S_i-successor, so A_i, together with the bool concepts implies D. This means that $A_i \sqcap \neg D$ is unsatisfiable. Suppose that we have to perform identification which leads to final clash. $Sat_\mathcal{T}$ contains concepts

$$W = E \sqcup (\leq nR.D) \qquad W_i = F_i \sqcup (\geq n_i S_i.A_i) \quad 1 \leq i \leq l$$

where $(\leq nR.D)$ is maximal in W, $(\geq n_i S_i.A_i)$ is maximal in W_i and each disjunct of E and F_i leads to clash in x. By the time a $\leq$-rule is applied, we have already

performed all possible $\bowtie$-rules, due to which the label of each S_i-successor contains either A_j or $\neg A_j$ for all $j \in \{1 \dots l\}$. According to Corollary 3, each creator concept is satisfiable and hence will remain satisfiable by taking its conjunction with either A_i or $\neg A_i$.

We use induction on l, the number of $\geq$-concepts to show that the assumption that the $\leq$-concept gives rise to final clash leads to contradiction.

The base case (of the second, inner induction) is when $l = 0$. There are no $\geq$-concepts in the label of x, so there are no involved successors to be identified.

We now turn to the inductive step (of the inner induction). We assume that if the label of x contains only $l' < l$ different $\geq$-concepts then the resulting successors can be identified into n nodes without clash.

1. In case $n_l > n$ then Rule3 is applicable on W and W_l, resulting in:

$$E \sqcup F_l \sqcup (\geq (n_l - n) S_l.dnf(A_l \sqcap \neg D))$$

   We know that $A_l \sqcap \neg D$ is unsatisfiable, so the DL calculus deduces $E \sqcup F_i$ (from Proposition 5). However, all disjuncts of E and F_i lead to clash in x, so we obtain a final clash even before introducing any $\leq$- and $\geq$-concept, contrary to our assumption.

2. If $n \geq n_l$, then concepts W and W_l are resolvable using Rule4, resulting in

$$E \sqcup F_l \sqcup (\leq (n - n_l) R.dnf(D \sqcap \neg A_l)) \sqcup (\geq 1 S_l.dnf(A_l \sqcap \neg D))$$

   Again, we know that $D \sqcap \neg A_l$ is unsatisfiable, so (from Proposition 5) the DL calculus deduces

$$E \sqcup F_l \sqcup (\leq (n - n_l) R.dnf(D \sqcap \neg A_l)) \tag{1}$$

   Due to the $\bowtie$-rule, the label of every successor contains either A_l or $\neg A_l$. $n - n_l < n$, so the inductive hypothesis holds for (1), i.e., all the successors whose label contains both D and $\neg A_l$ can be identified into $n - n_l$ nodes by deleting some successors that are not necessary. Further to this, there are n_l successors with creator concept A_l, plus some k other successors such that the $\bowtie$-rule put A_l into their labels.

   a) If $k \leq n_l$ then we can eliminate $n_l - k$ nodes from those having A_l as their creator concept, leaving exactly n_l successors whose label contains A_l. Contrary to our assumption, we obtain no final clash.

   b) If $k > n_l$ then each of the nodes whose creator concept is A_l can be eliminated since there are more then n_l other nodes satisfying A_l. All remaining successors originate from the $\geq$-concepts in $W_1 \dots W_{l-1}$. However, according to the inductive hypothesis (of the inner induction), these successors can be identified into n successors without clash.

This concludes the second inductive proof and the first one as well. We have showed that the assumption that a $\leq$-concept introduces inconsistency into the label of a node leads to contradiction. □

6.5 Conclusion

Let $\mathcal{T}$ be a $\mathcal{SHQ}$ TBox. Let $Sat_{\mathcal{T}}$ be the set of concepts obtained after performing preprocessing on $\mathcal{T}$ and then saturating it with the DL calculus. In the preceding subsections we have showed that if $Sat_{\mathcal{T}}$ does not contain $\bot$ then it is possible to build a model for $\mathcal{T}$ using the tableau algorithm. This concludes the proof of completeness for the DL calculus.

7 Towards a DL Calculus for ABox Reasoning

In this section we sketch an extension to the DL calculus that performs ABox reasoning. Although the answers that we obtained in our test queries are identical to those of other reasoners, we do not yet have a formal proof of completeness. Accordingly, this section should be seen as an indication of our future work. Currently, the DLog data reasoner uses the calculus described in [12] and the DL calculus is only in test phase.

We restrict our attention to *extensionally reduced* ABoxes, i.e., we assume that the concept assertions only contain literal concepts. An arbitrary knowledge base can be easily transformed to satisfy this constraint. Furthermore, we use the Unique Name Assumption (UNA): we assume that different names refer to different individuals. The rules in Figure 3 are added to the inference rules presented in Figure 1. As before, the resolved literals have to be maximal in their respective concepts.

$$\mathbf{5}\ \frac{C \sqcup (L_1 \sqcap L_2 \cdots \sqcap L) \qquad \neg L(a)}{C(a)} \qquad \mathbf{6}\ \frac{C \sqcup (L_1 \sqcap L_2 \cdots \sqcap L(a)) \qquad \neg L(a)}{C}$$

$$\mathbf{7}\ \frac{W \sqcup (\leq nR.C) \qquad \{R(a,b_i)\}_{i=1}^{n+1}}{W(a) \sqcup \bigsqcup_{i=1}^{n+1} \neg C(b_i)} \qquad \mathbf{8}\ \frac{W \sqcup (\leq nR.C)(a) \qquad \{R(a,b_i)\}_{i=1}^{n+1}}{W \sqcup \bigsqcup_{i=1}^{n+1} \neg C(b_i)}$$

Figure 3: ABox inference rules

An important property of the ABox DL calculus is that the rules for data axioms do not involve $\geq$-max concepts. This suggests that all inference steps involving $\geq$-max concepts can be performed *before accessing the ABox*, allowing us to break the ABox reasoning into two parts: (1) an ABox independent DL calculus is first applied to the TBox until all the consequences of $\geq$-max concepts are inferred; (2) next we perform the actual data reasoning using a much simpler TBox (as all $\geq$-max concepts can be eliminated). The output of the first phase is translated to first-order clauses during data reasoning and $\geq$-concepts give rise to skolem functions. Without $\geq$-max concepts, we can work with function-free clauses, which makes the reasoning task much easier [8].

8 Conclusion and Future Work

We have presented the DL calculus, a resolution based algorithm for deciding the consistency of a $\mathcal{SHQ}$ TBox. The novelty of this calculus is that it is defined directly on DL axioms. We showed that the algorithm is sound, complete and terminates. More work needs to be done to explore the real time complexity of the reasoning, as well as potential optimization techniques. We hope that further research will reveal that the DL calculus provides a reasonable alternative to the Tableau Method for certain reasoning tasks.

We have extended the DL calculus to consider ABox axioms as well, providing the basis of a two-phase ABox reasoning framework. The DL calculus is used to perform the first phase involving the TBox only. Given that the TBox is relatively stable over time, the speed of this phase is not crucial as it has to be performed only once. What really matters is that by the end of this phase we can eliminate many axioms that make the knowledge base much simpler. Thanks to the eliminations, we can translate the initial knowledge base into a set of first-order clauses that are function-free. The absence of function symbols enables us to use the query driven, highly efficient data reasoning techniques implemented in the DLog ABox reasoner. It has to be noted, however, that the ABox extension of the DL calculus requires further work.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F., editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2004.

[2] Bachmair, L. and Ganzinger, H. Resolution theorem proving. In Robinson, A. and Voronkov, A., editors, *Handbook of Automated Reasoning*, volume 1, chapter 2, pages 19–100. North Holland, 2001.

[3] Colmerauer, Alain and Roussel, Philippe. The birth of prolog. In Bergin, Jr., Thomas J. and Gibson, Jr., Richard G., editors, *History of programming languages—II*, pages 331–367. ACM, New York, NY, USA, 1996.

[4] Haarslev, V. and Möller, R. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004, Whistler, BC, Canada, June 2-5*, pages 163–173, 2004.

[5] Haarslev, V., Möller, R., van der Straeten, R., and Wessel, M. Extended Query Facilities for Racer and an Application to Software-Engineering Problems. In *Proceedings of the 2004 International Workshop on Description Logics (DL-2004), Whistler, BC, Canada, June 6-8*, pages 148–157, 2004.

[6] Horrocks, Ian, Sattler, Ulrike, and Tobies, Stephan. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999.

[7] Hustadt, Ullrich, Motik, Boris, and Sattler, Ulrike. Reasoning for Description Logics around SHIQ in a resolution framework. Technical report, FZI, Karlsruhe, 2004.

[8] Lukácsy, Gergely and Szeredi, Péter. Efficient description logic reasoning in Prolog: the DLog system. *Theory and Practice of Logic Programming*, 09(03):343–414, May 2009.

[9] Lukácsy, Gergely, Szeredi, Péter, and Kádár, Balázs. Prolog based description logic reasoning. In de la Banda, Maria Garcia and Pontelli, Enrico, editors, *Proceedings of 24th International Conference on Logic Programming (ICLP'08), Udine, Italy*, pages 455–469, December 2008.

[10] Motik, Boris. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany, January 2006.

[11] Sirin, Evren, Parsia, Bijan, Grau, Bernardo Cuenca, Kalyanpur, Aditya, and Katz, Yarden. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[12] Zombori, Zsolt. Efficient two-phase data reasoning for description logics. In Bramer, Max, editor, *IFIP AI*, volume 276 of *IFIP*, pages 393–402. Springer, 2008.

[13] Zombori, Zsolt and Lukácsy, Gergely. A resolution based description logic calculus. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik and Sattler, Ulrike, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford*, volume 477 of *CEUR Workshop Proceedings*, Oxford, UK, July 27-30 2009. `http://CEUR-WS.org/Vol-477` (accessed May 18, 2010).

Appendix: ALCHQ tableau rules

In this appendix we provide the rules of the tableau method. Even though the TBox reasoning starts out from an $\mathcal{SHQ}$ knowledge base, we quickly eliminate transitivity axioms during preprocessing and obtain an $\mathcal{ALCHQ}$ knowledge base. Accordingly, the rules provided in Figures 4 and 5 are those for the $\mathcal{ALCHQ}$ language. This appendix is not meant to explain how the tableau works. Instead, we provide it to make explicit what sorts of tableau rules we assume. For a comprehensive treatment of $\mathcal{SHIQ}$-tableau, we refer the reader to [6].

$\sqcap$-rule	
Condition:	$(C_1 \sqcap C_2) \in \mathcal{L}(x)$, x is not indirectly blocked and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$.
New state $\mathbf{T}'$***:***	$\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$.
$\sqcup$-rule	
Condition:	$(C_1 \sqcup C_2) \in \mathcal{L}(x)$, x is not indirectly blocked and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$.
New state $\mathbf{T}_1$***:***	$\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_1\}$.
New state $\mathbf{T}_2$***:***	$\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_2\}$.
$\exists$-rule	
Condition:	$(\exists R.C) \in \mathcal{L}(x)$, x is not blocked and x has no R-neighbour y for which $C \in \mathcal{L}(y)$.
New state $\mathbf{T}'$***:***	$V' = V \cup \{y\}$ ($y \notin V$ is a new node), $E' = E \cup \{(x, y, \})$, $\mathcal{L}'((x, y,)) = \{R\}$, $\mathcal{L}'(y) = \{C\}$.
$\forall$-rule	
Condition:	$(\forall R.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and x has an R-neighbour y for which $C \notin \mathcal{L}(y)$.
New state $\mathbf{T}'$***:***	$\mathcal{L}'(y) = \mathcal{L}(y) \cup \{C\}$.

Figure 4: The transformation rules of the $\mathcal{ALCHQ}$ tableau algorithm, part 1.

$\bowtie$ rule	
Condition:	$(\bowtie nR.C) \in \mathcal{L}(x)$, where $\bowtie$ is one of the symbols $\geq$ or $\leq$, x is not indirectly blocked, and x has an R-neighbour y for which $\{C, \sim C\} \cap \mathcal{L}(y) = \emptyset$.
New state $\mathbf{T}_1$***:***	$\mathcal{L}'(y) = \mathcal{L}(y) \cup \{C\}$.
New state $\mathbf{T}_2$***:***	$\mathcal{L}'(y) = \mathcal{L}(y) \cup \{\sim C\}$.
$\geq$-rule	
Condition:	$(\geq n\, R.C) \in \mathcal{L}(x)$, x is not blocked, and it is not the case that there exist nodes $y_1, \ldots, y_n$ such that no two of them are identifiable, and for every i, y_i is an R-neighbour of x, and $C \in \mathcal{L}(y_i)$ holds.
New state $\mathbf{T}'$***:***	$V' = V \cup \{y_1, \ldots, y_n\}$ ($y_i \notin V$ new nodes), $E' = E \cup \{(x, y_1,), \ldots, (x, y_n, \})$, $\mathcal{L}'((x, y_i,)) = \{R\}$, $\mathcal{L}'(y_i) = \{C\}$, for every $i = 1 \leq i \leq n$, $I' = I \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$.
$\leq$-rule	
Condition:	$(\leq n\, R.C) \in \mathcal{L}(x)$, x is not indirectly blocked, x has $n+1$ R-neighbours $y_0, \ldots, y_n$ such that $C \in \mathcal{L}(y_i)$ holds for every i, and there exist y_i and y_j that are identifiable.
For every $(0 \leq i < j \leq n)$, where y_i and y_j are identifiable, let $\{y, z\} = \{y_i, y_j\}$ so that x is not a successor of y:	
New state $\mathbf{T}_{ij}$***:***	$\mathcal{L}'(z) = \mathcal{L}(z) \cup \mathcal{L}(y)$, $\mathcal{L}'((x, y,)) = \emptyset$, $\mathcal{L}'((z, x,)) = \mathcal{L}((z, x,)) \cup \mathsf{Inv}(\mathcal{L}((x, y,)))$ if x is a successor of z, $\mathcal{L}'((x, z,)) = \mathcal{L}((x, z,)) \cup \mathcal{L}((x, y,))$ if x is not a successor of z, $I' = I[y \to z]$ (each occurrence of y is replaced by z).

Figure 5: The transformation rules of the $\mathcal{ALCHQ}$ tableau algorithm, part 2.

Acta Cybernetica **19** (2010) 591–605.

An Ontology Segmentation Tool*

András Simonyi† and Miklós Szőts†

Abstract

Extracting a minimal relevant segment of an extensive domain ontology is an often recurring problem in ontology engineering. We present a software solution to this problem that is the combination of an ontology-independent user interface generator and a module implementing an ontology segmentation algorithm. We describe the algorithm and compare it with other ontology segmentation methods proposed in the literature.

Keywords: ontology segmentation, ontology module extraction, user interface generation, cross-domain engineering

1 Introduction

The goal of the ImportNET project has been to develop an 'intelligent modular open source platform for intercultural and cross-domain SME networks' for the field of mechatronic design. According to the high level architecture, the general knowledge of the field is represented in a comprehensive ontology, the *reference ontology* in ImportNET terminology. When a new project is started, the relevant knowledge, represented in a so called *collaboration ontology*, is generated from the reference ontology by a software tool named *Ontology Integration Tool* (henceforth OIT).[1] The generation process consists of two phases:

1. Domain experts mark the important relevant/irrelevant components of the reference ontology, and make some minor modifications (add individuals, concepts etc.).

2. OIT generates a minimal (as far as the algorithm allows) subontology of the reference ontology that contains those pieces of knowledge that are related to the marked items.

*This work has been supported by ImportNET, a research and development project co-funded by the European Commission within the Sixth Framework Programme under Contract 033610, in the area of ICT for Networked Businesses.

†Applied Logic Laboratory, E-mail: {simonyi,szots}@all.hu

[1]This approach was first outlined in [9].

During the development of OIT we have devised a general method for the selection of the relevant subontology — the present paper offers an overview of this approach. Although in the ImportNET platform the generated collaboration ontology has to be transformed into an object oriented data model, we do not discuss this phase here.

2 Problem Analysis

There is a well known taxonomy of ontologies according to their generality: [6] introduced the notions of upper, domain, and application ontologies. We quote the characterisation of upper and domain ontologies from [12]:

> An upper ontology [...] is a high-level, domain-independent ontology, providing a framework by which disparate systems may utilise a common knowledge base and from which more domain-specific ontologies may be derived. The concepts expressed in such an ontology are intended to be basic and universal concepts to ensure generality and expressivity for a wide area of domains. (p. 2-2)

There are some well known upper ontologies; we use DOLCE [8]. Upper ontologies are in marked contrast to domain ontologies:

> A domain ontology specifies concepts particular to a domain of interest and represents those concepts and their relationships from a domain specific perspective. While the same concept may exist in multiple domains, the representations may widely vary due to the differing domain contexts and assumptions. [12, p. 2-3]

Finally, an application ontology is used in a software system. It represents a part of the domain knowledge that is relevant to a special task concerning the domain in question, and may be tuned to the requirements of the application.

Note that the above characterisations cannot be regarded as precise definitions, since their meaning depends on what we consider a domain. Moreover, there may exist ontologies that formalise only a small piece of knowledge concerning a domain, but are not connected to any software system — such an ontology would not fit into the above classification.

Clearly, if we have a domain ontology and an application ontology for the same domain, then the application ontology (with certain modifications) is a subontology of the domain ontology. Accordingly, the reference ontology of the ImportNET project can be considered a domain ontology and the collaboration ontology an application ontology.

In more general terms our problem can be formulated in the following way: how to obtain an application ontology from a domain ontology? In order to obtain an application ontology, firstly the intended application itself has to be specified. The simplest way of doing so is to mark some ontology items as relevant or irrelevant to

the application in question. In this case an appropriate application ontology will be a subontology of the domain ontology that

- contains the items marked relevant, but does not contain those marked irrelevant,
- contains those ontology items and pieces of information that are connected to the relevant items and are not marked irrelevant.

Of course, an *optimal* application ontology has to satisfy a further condition: it has to be minimal among the appropriate application ontologies in the sense that it cannot have a (proper) subontology which is also appropriate. The proper construal of the notion 'ontology items and pieces of information that are connected to the relevant items' is itself part of the problem. There are two important factors that must be taken into account by any solution of the problem: the restrictions contained by the ontology and the criterion of connectedness in the user's mind. While the first factor can be calculated automatically, the second has to be specified by the user. Consequently, the following two problems have to be addressed:

1. The naive user, who does not know the structure of the ontology, has to be able to select the relevant concepts and to parametrise the criterion of connectedness to be used, via a graphical user interface.
2. The concepts marked relevant and the connected ontology items have to be integrated into a quasi-optimal subontology which is an appropriate application ontology and a good approximation of an optimal one.

The next two sections present our solutions to these problems.

3 A Graphical User Interface for Naive Users

The need for generating user interfaces for ontologies is not new; see e.g. [2]. Paper [1] presents an ontology based portal where the interface is generated from a domain ontology and a small ontology describing GUI elements. Our problem is slightly different from theirs, because we aim at developing a general tool, which is ontology independent. In our solution (see Figure 1 on the following page) the specification of the interface is represented in an XML file called *design template*. A design template describes for every screen (state) of the user interface

- the GUI objects shown by it,
- how the data presented by the GUI objects can be obtained from the ontology,
- how to edit the ontology according to the user's activity,
- how to change the screens (states).

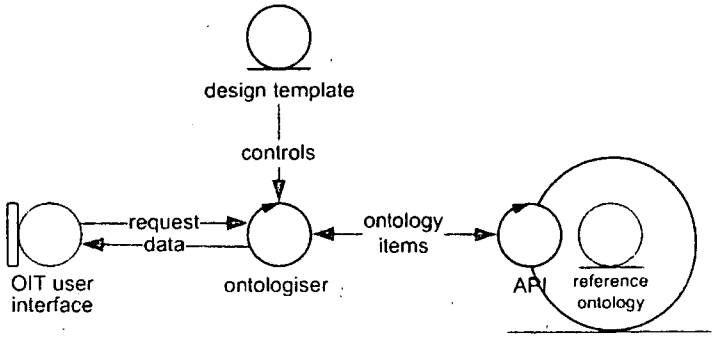

Figure 1: User interface architecture

The software module *Ontologiser* parses the design template, interprets it, and controls the operation of the whole program. The ontology, which is formulated in the OWL-DL ontology language, is stored in the memory of the Protégé editor [18] and accessed via Protégé's APIs. The user interface is totally thin: it only shows data received from the Ontologiser and receives data from the user, which is in turn transferred to the Ontologiser module.

A more detailed explanation of the structure of design templates and the operation of the Ontologiser module can be found in [17].

4 Integration Algorithm

A concept can be connected to a relevant concept in one or more of the following ways:

1. *Via a relevant ontology relation.* We discuss this type of connection, which we term 'relational connection', in the next subsection.

2. *By being a subconcept of the relevant concept and not marked irrelevant.* It can be assumed that if a subconcept of a relevant concept is irrelevant, then this fact is explicitly indicated by the user — in absence of such an indication, there is no reason to suppose that a subconcept can be left out from the collaboration ontology.

3. *By being a superconcept of the relevant concept and not marked irrelevant.* In contrast to the previous two connection types, superconcepts of a relevant concept are, in many cases, not specifically relevant to the collaboration. Nevertheless, they have to be included into the collaboration ontology in order to ensure that it forms a proper ontology with an appropriate upper ontology layer.

Starting from the concepts marked relevant by the user, the integration algorithm first has to find all concepts of the reference ontology that are relevant to the

collaboration by being connected to a relevant concept (either via a relational connection or the subconcept relationship). Secondly, the relevant fragment of the reference ontology has to be extended into a self-contained ontology by including the superconcepts of the relevant concepts.

4.1 Generating the Hull of a Concept

One of the key tasks of the sketched integration process is to find those concepts that are connected to a single relevant concept through relational connections. An ontology fragment containing just these concepts and the relevant concept in question is called the *hull* of the concept [9]. Before describing the integration algorithm, first we examine and make more precise the notions 'relational connection' and 'hull'.

4.1.1 Logical Analysis

There is a **relational connection** between two concepts A and B via relation R if and only if

(i) there may exist two individuals that are instances of A and B respectively and relation R holds between them, formally $\Diamond\exists x\exists y(A(x) \wedge B(y) \wedge R(x,y))$; and

(ii) the ordered pair $\langle A, B\rangle$ is minimal among those pairs of concepts that satisfy condition (i), that is, there are no concepts A' and B' such that A' is a subconcept of A, B' is a subconcept of B, $A' \neq A$ or $B' \neq B$, and $\Diamond\exists x\exists y(A'(x) \wedge B'(y) \wedge R(x,y))$.

The modality invoked in the definition is that of conceptual possibility or conceivability (from the user's point of view) — two ontology concepts are connected via a relation R if the user deems it conceptually possible that they have instances standing in relation R, and they also satisfy the minimality condition given by (ii).

The **hull** of a selected concept C is generated by forming the closure of the set $\{C\}$ under relational connections. The following is a simple illustration of the evolution of a concept's hull (see also Figure 2 on the next page):

Let A be a selected concept, let there be relational connections between A and $B_1, \ldots B_n$ via relations $R_1, \ldots, R_n$, respectively; and similarly relational connections between $E_1, \ldots, E_m$ and A via $Q_1, \ldots, Q_m$. At the first step concepts $B_1, \ldots, B_n$ and $E_1, \ldots, E_m$ make up the hull of A. The generation of the hull is iterative: in the same manner new concepts have to be added to $B_1, \ldots, B_n, E_1, \ldots, E_m$ and so on. Note that the inverses of the relations also have to be considered when relational connections are searched for. Clearly, relations $R_1, \ldots, R_n$ and $Q_1, \ldots, Q_m$ also belong to the hull.

Unfortunately, description logic does not allow expressing the notion of relational connection in the form we have defined above, and therefore we cannot determine precisely whether there is a relational connection between two concepts

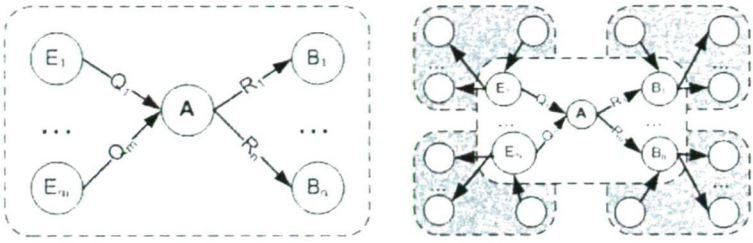

Figure 2: Generation of the hull of a concept

of an ontology that is formulated in a description logic language. Assuming that the class of concepts is closed under intersection, we can formulate a simple necessary condition: there may be a relational connection between two concepts A and B via relation R only if A is a subconcept of the domain of R, and B is a subconcept of the range of R. This condition is too permissive, since there may well be several subconcepts of the domain and range of a relation without a relational connection between them. This kind of situation often occurs when a relation is defined at a very high level. For instance, the PART_OF relation[2] often has the root concept of the ontology as its domain and range – e.g. in the DOLCE top ontology the domain and the range of PART_OF is the most general concept PARTICULAR. Nonetheless, certain restrictions provide clues that can help deciding whether there is a relational connection between two concepts.

Restrictions[3] on a concept A that *imply* the existence of some relational connections via R:

someValuesFrom	$\exists R.C$	There must be a relational connection between A and C via R.[4]
minCardinality	$\geq nR$	There must be a relational connection between A and one of the subconcepts of R's range via R.
hasValue	$R : i$	There must be a relational connection between A and the minimal concept(s) containing i via R.

Restrictions on a concept A that *exclude* the existence of some relational connections via R:

allValuesFrom	$\forall R.C$	C is the only subconcept of the range of R such that there is a relational connection between A and it via R.[5]
	$\forall R.\bot$	A has no relational connections via R.

[2] PART_OF(a, b) means that a is part of b.

[3] We consider only restrictions used in OWL.

[4] The implication holds only if the ontology is well axiomatised in the sense that C does not have a proper subconcept C' for which $\exists R.C'$ is true.

[5] Analogously to the case of minCardinality, the implication holds only if the ontology is reasonably axiomatised insofar as C does not have a proper subconcept C' for which $\forall R.C'$ is true (see previous footnote).

Relying on these conditions, we can define the maximal and minimal hull of a concept, which provide an 'upper bound' and a 'lower bound' of the concept's hull in the sense that the hull of a concept contains its minimal hull and is contained by its maximal hull. The **maximal hull** of a concept C is computed by forming the closure of $\{C\}$ under the relation which holds between two concepts A and B iff there is a relation R such that A is a subconcept of R's domain and B is a subconcept of R's range, while the **minimal hull** is calculated by forming the closure of $\{C\}$ under the relation which holds between A and B iff there is an R for which the ontology contains a restriction on A that *implies* that A and B is connected via R.

4.1.2 Heuristic Rules

Although both the minimal and maximal hull of a concept can be precisely determined in the case of a description logic based ontology, in practice we look for an ontology fragment which is in between the minimal and maximal hulls, as the maximal hull is frequently too large — sometimes including the whole reference ontology — while the minimal hull may leave out relevant concepts whose relevance is not declared explicitly by restrictions. To find an appropriate fragment, the knowledge of the ontology engineer or user has to be relied upon — knowledge which is often not expressible in the ontology language, but nevertheless can be utilised for checking the extension of hulls in the form of heuristic rules.

We have considered two types of such knowledge:

- the meaning of a relation makes it superfluous to consider its domain or range — this makes it possible to use relation filtering in OIT;
- the user may provide some information in the selection phase that may control the selection of relational connections.

The first case can be clarified using the example of the PART_OF relation. Knowledge concerning the parts of an object may surely be relevant if the object in question is relevant. Consequently, if the selected concept is included in the range of the PART_OF relation, then its domain has to be included into its hull. However, even if something is relevant for a collaboration, it cannot be said that everything that contains it as a part is also relevant. Accordingly, if the concept is in the domain of the PART_OF relation, the range may be irrelevant, and it may not get into the hull. Since the PART_OF relation is transitive, this means that only a segment of a chain of concepts is added to the hull, as Figure 3 shows.

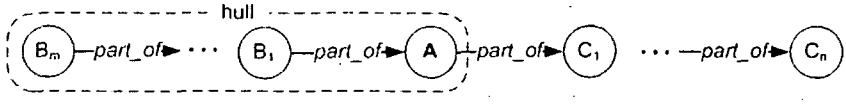

Figure 3: An example of directed relations

There can be other relations whose domain or range may be irrelevant -- we will call relations belonging to this class **directed relations**. They have to be collected from the actual reference ontology, and it has to be decided in which direction they are irrelevant. In our implementation this information is described in a parameter file.

We have introduced two ways of manual control, both provided in the selection (customisation) phase: global and local manual control. In both cases some relations are marked as irrelevant for the hull generation process.

Global manual control influences the hull generation in a completely general manner: the user may select some relations that are not to be considered in the generation of the hull at all.

In the case of **local manual control** the user may mark certain relational connections of some concepts as irrelevant, and these connections are not considered when generating the hull. The user interface outlined in the previous section helps the user to select these relational connections.

In general, global and local manual control can help to generate a quasi-optimal hull. However, certain relations require special ways of handling. We have used the quality/quale structure of the DOLCE upper ontology [8] to model the properties of concepts. In this case subrelations of the relation HAS-QUALITY have to be considered only if the necessary conditions of relational connections hold. At the same time, when a quality concept is included, the corresponding quale concept has to be included as well.

4.2 The Algorithm

The collaboration ontology is generated in a complicated system of iterations. During this process we do not build a new ontology, but mark the concepts and relations to be included in the collaboration ontology with labels. The collaboration ontology as a new ontology is generated only when the user indicates that the process has been finished. This solution allows saving the labelled reference ontology, and using it again as a starting point when a new project is built upon the previous one.

Concepts can enter the collaboration ontology in a number of different ways:

- the user selects them,
- they are in the hull of an included concept,
- they are subconcepts of an included concept,
- they are superconcepts of an included concept.

In the last three cases the concepts are included by the integration module of the program. Although these cases are handled by three different procedures, they have to be organised into an iterative process, since the subconcepts of the concepts generated as elements of a hull also have to be generated, and the hulls of subconcepts have to be included as well. As we have already noted, the hulls of superconcepts

do not have to be generated; the superconcepts are needed only to form a complete ontology from the generated concepts. In order to control the iteration, the included concepts are collected into a list (called CONCEPT_TO_BE_PROCESSED), and it is marked whether the 'generation of hull' and 'generation of subconcepts' steps of the algorithm (see below) have already been executed on them.

The process of generating the collaboration ontology consists of the following steps:

0 The process is prepared by collecting the highest selected concepts into the list CONCEPT_TO_BE_PROCESSED.

1 The superconcepts of concepts in the list CONCEPT_TO_BE_PROCESSED are generated.

2 A cycle is started and runs as long as there are unprocessed concepts in CONCEPT_TO_BE_PROCESSED as follows:

  2.1 The hulls of those concepts in CONCEPT_TO_BE_PROCESSED that have not yet been processed by the procedure GENERATION_OF_HULLS are generated. In parallel with the labelling of ontology items, the generated concepts are added to the list CONCEPT_TO_BE_PROCESSED. The processed concepts are marked as 'processed by the procedure GENERATION_OF_HULLS.'

  2.2 The subconcepts of those concepts in the list CONCEPT_TO_BE_PROCESSED that have not yet been processed by the procedure GENERATION_OF_SUBCONCEPTS are generated. In parallel with the labelling of ontology items, the generated concepts are added to the list CONCEPT_TO_BE_PROCESSED. The processed concepts are marked as 'processed by the procedure GENERATION_OF_SUBCONCEPTS.'

  2.3 If there are no further unprocessed concepts in the list CONCEPT_TO_BE_PROCESSED then the cycle is finished.

3 The instances of concepts in the list CONCEPT_TO_BE_PROCESSED are generated (GENERATION_OF_INSTANCES).

4 The highest concepts in the newly labelled ontology are generated and marked as processed.

5 The superconcepts of the highest concepts in the list CONCEPT_TO_BE_PROCESSED are generated by the procedure GENERATION_OF_SUPERCONCEPTS.

Note that only the highest selected concepts are considered in the starting step, since the subconcepts of every selected concept are generated automatically.

The main danger of automatic generation is that some very high level concepts get into the collaboration ontology, and the further iteration steps include almost the whole ontology. Step 1 is placed before the iteration in order to avoid this

problem. Nonetheless, the effectiveness of the algorithm is highly dependent on the structure of the source ontology — only a suitably axiomatised and coherent ontology can be processed with good result.

5 A Walk-Through Example

In this section we illustrate the operation of OIT with a simple example, in which the user selects only one electronic component. The purpose of the example is to show the relationship between components and their functions, to illustrate the 'undesired side effects' the integration algorithm might have, and to indicate how the resulting segment can be improved upon by changing some of the input parameters.

Let us suppose that the user selects the ontology concept MICROCONTROLLER as relevant (this makes it likely that she considers the siblings of MICROCONTROLLER to be irrelevant, since otherwise she would have selected some of the siblings as well). Figure 4 on the next page shows a part of the segmentation's result, that is, the components that are included into the resulting segment.

As can be seen, the subconcepts and superconcepts of the selected concept are included; however, the inclusion of the concept CONNECTOR may seem at first sight puzzling. For an explanation let us consider a small fragment of the reference ontology (Figure 5 on page 602). In the first step of making the hull concepts CONTROL, DETECT, and CHANNEL are included. Note that the 'forAll' restriction restricts the concepts included from the range of relation TYPICALLY-HAS-FUNCTION. However, in the second phase, when the hull of concept CHANNEL is generated, the concept CONNECTOR is included because of the relation TYPICALLY-FUNCTION-OF.

Note that if the inverse of the relation TYPICALLY-HAS-FUNCTION was unnamed, CONNECTOR would still be included, since the generation of the hull considers the inverses of relations as well (see Figure 2 on page 596). If the user looks at the result and decides that CONNECTOR is irrelevant and should not be included, then she may unselect it, and initiate another execution of the segmentation algorithm. In this new session the inclusion of concept CONNECTOR and its subconcepts will be prevented.

6 Related Work

The problem of ontology segmentation and modularisation had received relatively little attention until the last few years, when comprehensive studies of ontology segmentation and modularisation were conducted within important semantic web projects such as WonderWeb [15] and Knowledge Web [13], and a number of different approaches to automatic and semi-automatic ontology segmentation have emerged in the literature.

The proposed solutions can be classified into three broad categories. There are graph-theoretic approaches (e.g. [16, 4]), which transform the input ontology into a directed graph and utilise general network-theoretic algorithms to find minimal

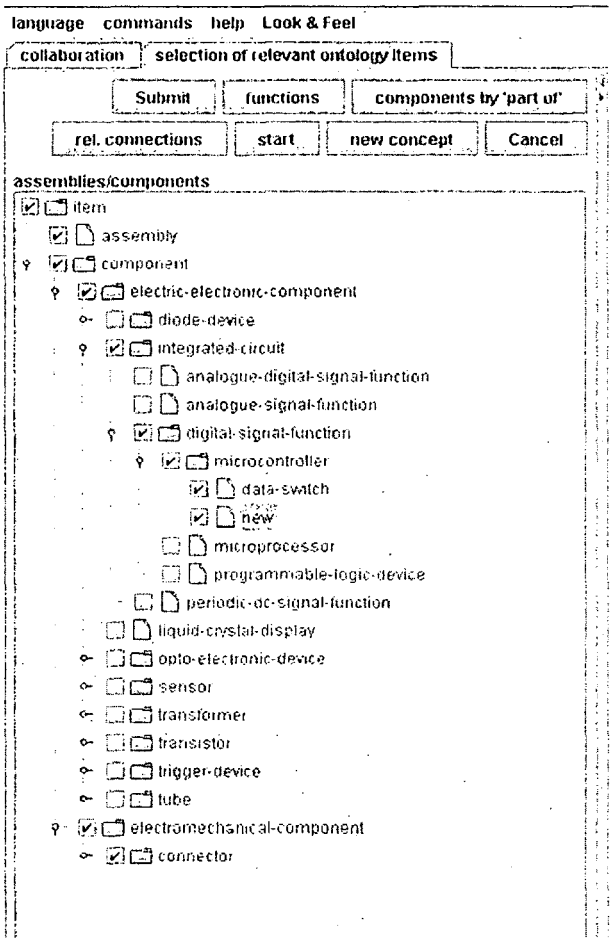

Figure 4: A segmentation result (concept NEW was added by the user)

subgraphs that contain the input items and meet certain abstract conditions of connectedness; model-theoretic proposals, which construe the criterion of relevance in terms of semantic consequence (e.g. [5, 7]), and, finally, heuristic solutions seeking to find the relevant subontology on the basis of certain special relationships between concepts (e.g. different types of relational connections, role in a quality-quale structure, mereological relationships etc.) that are treated individually, according to their semantics (e.g. [11, 3]). Our approach belongs to the third category.

A comparison of different ontology segmentation methods has to keep in view the purpose of the segmentation to be performed. In our case the generated ontology fragment has to serve as the knowledge base of a software system — accordingly, both the criteria of relevance and the requirements towards the generated ontol-

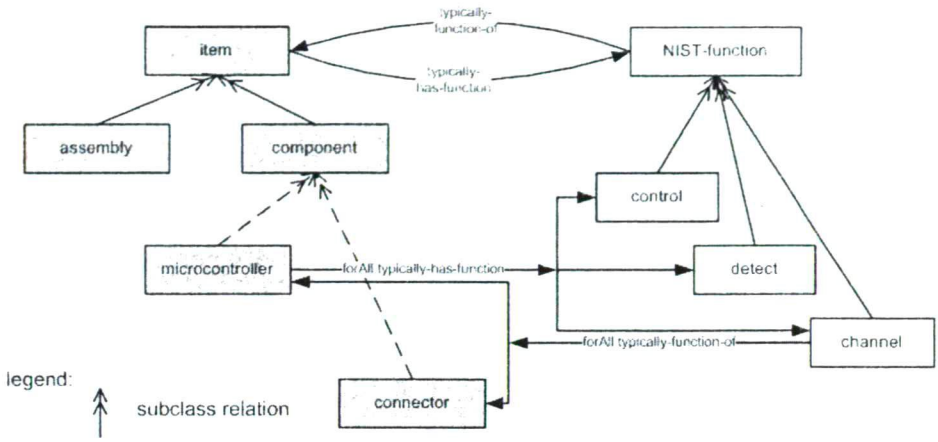

Figure 5: A fragment of the reference ontology

ogy differ from those arising in the case of other applications, e.g. in an ontology search system. Firstly, there are determinate — although often not explicit — requirements and criteria of correctness. It follows that abstract, e.g. graph-theoretic methods cannot be applied without extensive modifications and 'customization', because the segmentation process has to take into account the semantics of relations e.g. that of HAS-QUALITY or HAS-QUALE. Moreover, the input ontology cannot be treated simply as a directed graph since inverse relations have to be considered even if they are anonymous. Our case is also complicated by the fact that the task is not simple concept hull generation, because the input contains several concepts.

We compare our approach with the heuristic segmentation solution of [11], which is the most similar to our proposal. The method described by [11] generates the required ontology segment by computing the closure of the selected concepts under relational connections and the subconcept/superconcept relationship,[6] and tries to limit the size of the resulting segment both by filtering out certain relational connections on semantic grounds, and by limiting the distance of the included concepts from the initially selected ones.

In our case the fulfilment of the relevance criteria is ensured by the cooperation of automatic processes and the iterated manual intervention of the user. The important role played by the users' interaction with the system is due to the fact that in the ImportNET use case there is no fault tolerance: the resulting collaboration ontology has to contain each and every relevant ontology item, since it has to be transformed into an object oriented data model. As a consequence, the distance-based pruning method described in [11] has been unusable in ImportNET, in spite of the fact that it occurs in early descriptions of the platform (e.g. in [9]).

Another important difference lies in the special role of superconcepts. While

[6]Closure under the subconcept relation is partial, because only the initally selected concepts' subconcepts are included into the segment.

the elements of the hull and the subconcepts participate in the later iterations in the sense that further subconcepts and hulls have to be generated from them, the same does not apply to superconcepts. This feature of the algorithm solves the problem which motivated the introduction of distance limits in [11]. Our approach generates the hull of a concept on the basis of the notion of relational connection. In contrast, [11] treats restrictions as superconcepts, which are independent from the 'property filtering' function of the system i.e. from the domains and ranges of relations.

Finally, in our approach the user has an important role to play in the generation of concept hulls, therefore she can specify the relevance of individual relational connections depending on the concrete task at hand.

7 Conclusion and Perspectives

This paper has presented an ontology segmentation tool. The objective is to select a subontology of a domain ontology that is relevant to a given problem. The criterion of relevance is supplied by the end users by selecting relevant/irrelevant ontology elements (mainly concepts, but relations and relational connections can also be selected). The process of segmentation consists of a high level iteration: selection of relevant items by the users and automatic segmentation follow each other.

The end users' interaction with the system raises an important problem: the ontology has to be handled by naive users, who do not know anything about ontologies. Accordingly, a separate module of the system bridges the gap between ontology structure and the end users' business logic. The module is based on a general method for providing ontology editing tools for naive users, where the specification of the user interface is given in an XML file.

The ontology segmentation process consists of three activities: generating the hull of a concept, generating subconcepts, and generating superconcepts. The steps of generating the hulls and subconcepts of concepts are organised into an iteration. Superconcepts are included into the generated segment only to make it a coherent ontology; neither their hulls, nor their subclasses are generated.

The most important differences between our approach to the problem of ontology segmentation and other proposed solutions are due to the fact that in our use case the generated ontology segment has to be transformed into an object oriented data model, and, consequently, we have no fault tolerance. This is why the users have an exceptionally important role in the segmentation process.

The tool presented in the paper is in a prototype version, and has been used with a good result in the ImportNET project. However, further experiments are needed with different kinds of ontologies to develop it into a generally usable tool.

OIT consists of two well separated parts: a module making it possible for naive users to edit ontologies in a restricted way, and the segmentation module. These parts can be used in different scenarios and for different goals. At the end of the ImportNET project a new scenario has been outlined (see [10]), which connects functional design [14] with the generation of a collaboration ontology. This may help

the development of an up-to-date design technology in mechatronic engineering.

Acknowledgement. The authors are grateful to the ImportNET community for the inspiration and support they provided, and to two anonymous reviewers for constructive comments.

References

[1] Blaszczynski, J., Kosiedowski, M., Mazurek, C., and Wilk, S. Ontologies for Knowledge Modeling and Creating User Interface in the Framework of Telemedical Portal. In Stormer, H., Meier, A., and Schumacher, M., editors, *Proceedings of the ECEH'06, Fribourg, Switzerland, October 12-13*, pages 275–286, Bonn, 2006. Gesellschaft für Informatik.

[2] Bojars, U. Captsolo Weblog: Ontologies => User Interface, May 17, 2004. WWW document. `http://captsolo.net/info/blog_a.php/2004/05/17/p520` (accessed May 18, 2010).

[3] d'Aquin, D., Sabou, D., and Motta, P. Modularization: A Key for The Dynamic Selection of Relevant Knowledge Components. In *WoMO'06 Workshop on Modular Ontologies*, 2006. `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-232/paper2.pdf` (accessed May 18, 2010).

[4] Doran, P., Tamma, V., and Iannone, L. Ontology Module Extraction for Ontology Reuse: An Ontology Engineering Perspective. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 61–70, New York, 2007. ACM Press.

[5] Grau, B.C., Horrocks, I., Kazakov, Y., and Sattler, U. Just the Right Amount: Extracting Modules from Ontologies. In *Proceedings of the 16th International Conference on World Wide Web*, pages 717–726, New York, 2007. ACM Press.

[6] Guarino, N. Formal Ontology and Information Systems. In Guarino, N., editor, *Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy*, pages 3–15, Amsterdam, 1998. IOS Press.

[7] Konev, B., Lutz, C., Walther, D., and Wolter, F. Semantic Modularity and Module Extraction in Description Logics. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., and Avouris, N., editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 55–59, Amsterdam, 2008. IOS Press.

[8] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. WonderWeb Deliverable D18: Ontology Library (final), 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf` (accessed May 18, 2010).

[9] Ovtcharova, J., Mahl, A., and Krikler, R. Approach for a Rule Based System for Capturing and Usage of Knowledge in the Manufacturing Industry. In Wang, K., Kovacs, G., Wozny, M., and Fang, M., editors, *Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management*, pages 134–143, Boston, 2006. Springer.

[10] Ovtcharova, J., Marinov, M., Szőts, M., Simonyi, A., and Schubert, P. Ontology-Based, Function Oriented Support for Cross-Domain Engineering. In preparation.

[11] Seidenberg, J. and Rector, A. Web Ontology Segmentation: Analysis, Classification and Use. In *Proceedings of the 15th International Conference on World Wide Web*, pages 13–22, New York, 2006. ACM Press.

[12] Semy, S. K., Pulvermacher, M. K., and Obrst, L. J. Toward the Use of an Upper Ontology for U.S. Government and U.S. Military Domains: An Evaluation. Technical report, MITRE, 2004. `http://www.mitre.org/work/tech_papers/tech_papers_05/04_1175/04_1175.pdf` (accessed May 18, 2010).

[13] Spaccapietra, S., editor. Knowledge Web Deliverable 2.1.3.1: Report on Modularization of Ontologies, 2003. `http://wasp.cs.vu.nl/knowledgeweb/Deliverables/D2.1.3.1/D2.1.3.1-Modularization.pdf` (accessed May 18, 2010).

[14] Stone, R. B. and Wood, K. L. Development of a Functional Basis for Design. *Journal of Mechanical Design*, 122:359–370, 2000.

[15] Stuckenschmidt, H. and Klein, M. Modularization of Ontologies: WonderWeb Deliverable D21, 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D21.pdf` (accessed May 18, 2010).

[16] Stuckenschmidt, H. and Klein, M. Structure-based Partitioning of Large Concept Hierarchies. In *The Semantic Web — ISWC 2004*, pages 289–303, Berlin, 2004. Springer.

[17] Szőts, M. and Schneider, F. Generating Ontology User Interface for Naïve Users. In *International Conference on Collaborative Mechatronic Engineering (ICCME'09), Salzburg, Austria*, 2009. `http://importnet.salzburgresearch.at/images/ImportNET_Bilder/Paper/session_ii_szots_schneider.pdf` (accessed May 18, 2010).

[18] The Protégé Ontology Editor and Knowledge Acquisition System. WWW document. `http://protege.stanford.edu` (accessed May 18, 2010).

Regular Papers

Acta Cybernetica **19** (2010) 609–634.

Petri Net Controlled Grammars with a Bounded Number of Additional Places*

Jürgen Dassow† and Sherzod Turaev‡

Abstract

A context-free grammar and its derivations can be described by a Petri net, called a *context-free Petri net*, whose places and transitions correspond to the nonterminals and the production rules of the grammar, respectively, and tokens are separate instances of the nonterminals in a sentential form. Therefore, the control of the derivations in a context-free grammar can be implemented by adding some features to the associated cf Petri net. The addition of new places and new arcs from/to these new places to/from transitions of the net leads grammars controlled by *k-Petri nets*, i.e., Petri nets with additional k places. In the paper we investigate the generative power and give closure properties of the families of languages generated by such Petri net controlled grammars, in particular, we show that these families form an infinite hierarchy with respect to the numbers of additional places.

Keywords: grammars, grammars with regulated rewriting, Petri nets, Petri net controlled grammars

1 Introduction

It is well-known fact that context-free grammars are not able to cover all phenomena of natural and programming languages, and also with respect to other applications of sequential grammars they cannot describe all aspects. On the other hand, context-sensitive grammars are powerful enough but have bad features with respect to decidability problems which are undecidable or at least very hard. Therefore it is a natural idea to introduce grammars which use context-free rules and have a device which controls the application of the rules. The monograph [2] gives a summary of this approach.

*This paper is an extended version of the paper presented at the Second International Conference on Language and Automata Theory and Applications, March 13-19, 2008, Tarragona, Spain [3].

†Otto-von-Guericke-Universität Magdeburg, PSF 4120, D-39016 Magdeburg, Germany, E-mail: dassow@iws.cs.uni-magdeburg.de

‡Faculty of Computer Science and Information Technology, UPM, 43400 Serdang, Selangor, Malaysia, E-mail: sherzod@fsktm.upm.edu.my

A context-free grammar and its derivation process can be described by a Petri net where places correspond to nonterminals, transitions are the counterpart of the productions, the tokens reflect the occurrences of symbols in the sentential form, and there is a one-to-one correspondence between the application of (sequences of) rules and the firing of (sequence of) transitions (see, [1]). Therefore it is a natural idea to control the derivations in a context-free grammar by adding some features to the associated Petri net.

In [7] and [13] it has been shown that by adding some places and arcs which satisfy some structural requirements one can generate well-known families of languages as random context languages, vector languages and matrix languages. Thus the control by Petri nets can be considered as a unifying approach to different types of control (note that random context is a control by occurrence/non-occurrence of letters whereas matrices give a prescribed set of sequences in which the productions have to be applied). In this paper we add new places, called *counters*, and new arcs associated with the new places. Adding k places leads to a control by k-Petri nets. The aim of this paper is the study of properties of the family of languages which can be generated by context-free grammars with a control by k-Petri nets. We present results on the generative power and we give some closure properties.

The paper is organized as follows. In Section 2 we give some notions and definitions from the theories of formal languages and Petri nets needed in the sequel. Moreover we introduce the Petri net associated with a context-free grammar. In Section 3 we construct the new Petri net control mechanism and define the corresponding grammar. Furthermore, we give some examples. In Section 4 we show that context-free grammars with the simple control by one additional place can generate non-context-free languages. We also give relations to valence grammars and vector grammars. Furthermore, we show that we get an infinite hierarchy with respect to the numbers of additional places. In Section 5 we investigate the fundamental closure properties of the families of languages generated by k-Petri net controlled grammars.

2 Preliminaries

The reader is assumed to be familiar with basic notions of formal language theory and Petri net theory as, e.g. contained in [8, 2, 4, 5, 6, 9, 10, 11, 12].

2.1 Grammars

Let Σ be an *alphabet* which is a finite nonempty set of symbols. A *string* over the alphabet Σ is a finite sequence of symbols from Σ. The *empty* string is denoted by λ. The set of all strings over the alphabet Σ is denoted by Σ^*. A subset of Σ^* is called a *language*. The *length* of a string w, denoted by $|w|$, is the number of occurrences of symbols in w. The number of occurrences of a symbol a in a string w is denoted by $|w|_a$. For a subset Δ of Σ, the number of occurrences of symbols of Δ in a string $w \in \Sigma^*$ is denoted by $|w|_\Delta$.

The operation *shuffle* for languages $L_1, L_2 \subseteq \Sigma^*$ is defined by

$$\begin{aligned}\mathrm{Shuf}(L_1, L_2) = \{&u_1 v_1 u_2 v_2 \cdots u_n v_n \mid u_1 u_2 \cdots u_n \in L_1, v_1 v_2 \cdots v_n \in L_2,\\ &u_i, v_i \in \Sigma^*, 1 \le i \le n\}\end{aligned}$$

and for $L \subseteq \Sigma^*$,

$$\begin{aligned}\mathrm{Shuf}^1(L) &= L,\\ \mathrm{Shuf}^k(L) &= \mathrm{Shuf}(\mathrm{Shuf}^{k-1}(L), L), k \ge 2,\\ \mathrm{Shuf}^*(L) &= \bigcup_{k \ge 1} \mathrm{Shuf}^k(L).\end{aligned}$$

A *context-free grammar* is a quadruple $G = (V, \Sigma, S, R)$ where V and Σ are the disjoint finite sets of *nonterminal* and *terminal* symbols, respectively, $S \in V$ is the *start* symbol and $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of *(production) rules*. Usually, a rule (A, x) is written as $A \to x$. A rule of the form $A \to \lambda$ is called an *erasing rule*. $x \in (V \cup \Sigma)^+$ *directly derives* $y \in (V \cup \Sigma)^*$, written as $x \Rightarrow y$, iff there is a rule $r = A \to \alpha \in R$ such that $x = x_1 A x_2$ and $y = x_1 \alpha x_2$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A derivation using the sequence of rules $\pi = r_1 r_2 \cdots r_n$ is denoted by $\stackrel{\pi}{\Rightarrow}$ or $\stackrel{r_1 r_2 \cdots r_n}{\Longrightarrow}$. The *language* generated by G is defined by $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$. The family of context-free languages is denoted by **CF**.

A *vector grammar* is a quadruple $G = (V, \Sigma, S, M)$ where V, Σ, S are defined as for a context-free grammar, and M is a finite set of strings over a set of context-free rules called *matrices*. The language generated by the grammar G is defined by $L(G) = \{w \in \Sigma^* \mid S \stackrel{\pi}{\Rightarrow} w \text{ and } \pi \in \mathrm{Shuf}^*(M)\}$.

An *additive valence grammar* is a quintuple $G = (V, \Sigma, S, R, v)$ where V, Σ, S, R are defined as for a context-free grammar and v is a mapping from R into the set $\mathbb{Z}$ of integers. The language generated by G consists of all strings $w \in \Sigma^*$ such that there is a derivation $S \stackrel{r_1 r_2 \cdots r_n}{\Longrightarrow} w$ where $\sum_{i=1}^{n} v(r_i) = 0$.

A *positive valence grammar* is a quintuple $G = (V, \Sigma, S, R, v)$ whose components are defined as for an additive valence grammar. The language generated by G consists of all strings $w \in \Sigma^*$ such that there is a derivation $S \stackrel{r_1 r_2 \cdots r_n}{\Longrightarrow} w$ where $\sum_{i=1}^{n} v(r_i) = 0$ and for any $1 \le j < n$, $\sum_{i=1}^{j} v(r_i) \ge 0$.

The families of languages generated by vector, additive valence and positive valence grammars (with erasing rules) are denoted by **V**, **aV** and **pV** ($\mathbf{V}^\lambda$, $\mathbf{aV}^\lambda$ and $\mathbf{pV}^\lambda$), respectively.

2.2 Petri Nets

A *Petri net* (PN) is a construct $N = (P, T, F, \phi)$ where P and T are disjoint finite sets of *places* and *transitions*, respectively, $F \subseteq (P \times T) \cup (T \times P)$ is the set of *directed arcs*, $\phi : (P \times T) \cup (T \times P) \to \{0, 1, 2, \cdots\}$ is a *weight function*, where $\phi(x, y) = 0$ for all $(x, y) \in ((P \times T) \cup (T \times P)) - F$. A Petri net can be represented

by a bipartite directed graph with the node set $P \cup T$ where places are drawn as *circles*, transitions as *boxes* and arcs as *arrows*. The arrow representing an arc $(x, y) \in F$ is labeled with $\phi(x, y)$; if $\phi(x, y) = 1$, the label is omitted.

A mapping $\mu : P \to \{0, 1, 2, \ldots\}$ is called a *marking*. For each place $p \in P$, $\mu(p)$ gives the number of *tokens* in p. Graphically, tokens are drawn as small solid *dots* inside circles. ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$ are called *pre-* and *post-sets* of $x \in P \cup T$, respectively. For $X \subseteq P \cup T$, define ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. For $t \in T$ ($p \in P$), the elements of ${}^\bullet t$ (${}^\bullet p$) are called *input* places (transitions) and the elements of $t^\bullet$ ($p^\bullet$) are called *output* places (transitions) of the transition t (the place p).

A transition $t \in T$ is *enabled* by marking μ if and only if $\mu(p) \geq \phi(p, t)$ for all $p \in P$. In this case t can *occur* (*fire*). Its occurrence transforms the marking μ into the marking μ' defined for each place $p \in P$ by $\mu'(p) = \mu(p) - \phi(p, t) + \phi(t, p)$. We write $\mu \xrightarrow{t} \mu'$ to indicate that the firing of t in μ leads to μ'. A finite sequence $t_1 t_2 \cdots t_k$, $t_i \in T, 1 \leq i \leq k$, is called *an occurrence sequence* enabled at a marking μ and finished at a marking μ' if there are markings $\mu_1, \mu_2, \ldots, \mu_{k-1}$ such that $\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \ldots \xrightarrow{t_{k-1}} \mu_{k-1} \xrightarrow{t_k} \mu'$. In short this sequence can be written as $\mu \xrightarrow{t_1 t_2 \cdots t_k} \mu'$ or $\mu \xrightarrow{\nu} \mu'$ where $\nu = t_1 t_2 \cdots t_k$.

A *marked* Petri net is a system $N = (P, T, F, \phi, \iota)$ where (P, T, F, ϕ) is a Petri net, ι is the *initial marking*. Let M be a set of markings, which will be called *final* markings. An occurrence sequence ν of transitions is called *successful* for M if it is enabled at the initial marking ι and finished at a final marking τ of M. If M is understood from the context, we say that ν is a successful occurrence sequence.

2.3 Context-Free Petri Nets

The construction of the following type of Petri nets is based on the idea of using similarity between the firing of a transition and the application of a production rule in a derivation in which places are nonterminals and tokens are different occurrences of nonterminals.

Definition 1. *A context-free Petri net (in short, a cf Petri net) with respect to a context-free grammar $G = (V, \Sigma, S, R)$ is a tuple $N = (P, T, F, \phi, \beta, \gamma, \iota)$ where*

- *(P, T, F, ϕ) is a Petri net;*
- *labeling functions $\beta : P \to V$ and $\gamma : T \to R$ are bijections;*
- *there is an arc from place p to transition t if and only if $\gamma(t) = A \to \alpha$ and $\beta(p) = A$. The weight of the arc (p, t) is 1;*
- *there is an arc from transition t to place p if and only if $\gamma(t) = A \to \alpha$ and $\beta(p) = x$ where $|\alpha|_x > 0$. The weight of the arc (t, p) is $|\alpha|_x$;*
- *the initial marking ι is defined by $\iota(\beta^{-1}(S)) = 1$ and $\iota(p) = 0$ for all $p \in P - \{\beta^{-1}(S)\}$.*

We also use the natural extension of the labeling function $\gamma : T^* \to R^*$, which is done in the usual manner.

Example 1. Let G_1 be a context-free grammar with the rules:

$$r_0 : S \to AB,\ r_1 : A \to aAb,\ r_2 : A \to ab,\ r_3 : B \to cB,\ r_4 : B \to c$$

(the other components of the grammar can be seen from these rules). Figure 1 illustrates a cf Petri net N with respect to the grammar G_1. Obviously,

$$L(G_1) = \{a^n b^n c^m \mid n, m \geq 1\}.$$

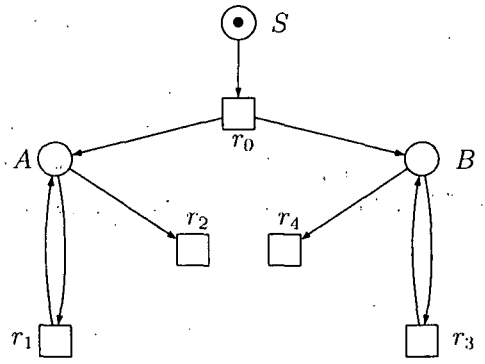

Figure 1: A cf Petri net N

The following proposition shows the similarity between terminal derivations in a context-free grammar and successful occurrences of transitions in the corresponding cf Petri net.

Proposition 1. *Let $N = (P, T, F, \phi, \iota, \beta, \gamma)$ be the cf Petri net with respect to a context-free grammar $G = (V, \Sigma, S, R)$. Then $S \overset{r_1 r_2 \cdots r_n}{\Longrightarrow} w, w \in \Sigma^*$, is a derivation in G iff $t_1 t_2 \cdots t_n$, $\iota \xrightarrow{t_1 t_2 \cdots t_n} \mu_n$, is an occurrence sequence of transitions in N such that $\gamma(t_1 t_2 \cdots t_n) = r_1 r_2 \cdots r_n$ and $\mu_n(p) = 0$ for all $p \in P$.*

Proof. Let $S \overset{r_1 r_2 \cdots r_n}{\Longrightarrow} w, w \in \Sigma^*$ be a derivation in the grammar G. By induction on the number $1 \leq k \leq n$ of derivation steps, we show that $t_1 t_2 \cdots t_n$ with $\gamma(t_1 t_2 \cdots t_n) = r_1 r_2 \cdots r_n$ is an occurrence sequence enabled at ι and finished at the marking μ_n where $\mu_n(p) = 0$ for all $p \in P$.

Let $k = 1$. $S \Rightarrow_{r_1} w_1$, i.e., the sentential form w_1 is obtained from S by the application of a rule $r_1 : S \to w_1 \in R$. Then the transition $t_1 = \gamma^{-1}(r_1)$ also occurs as its input place $\beta^{-1}(S)$ has a token, i.e., by definition, $\iota(\beta^{-1}(S)) = 1$. Let $\iota \xrightarrow{t_1} \mu_1$. Then for each $A \in V$, we have $\mu_1(p) = |w_1|_A$ where $p = \beta^{-1}(A)$.

Suppose $S \xRightarrow{r_1 r_2 \cdots r_m} w_m, w_m \in (V \cup \Sigma)^*$, $1 \leq m \leq k-1 < n$, and $t_1 t_2 \cdots t_m$ be an occurrence sequence of transitions of N such that $\gamma(t_1 t_2 \cdots t_m) = r_1 r_2 \cdots r_m$. Consider case $m = k$. Then the transition $t_k = \gamma^{-1}(r_k)$, $r_k : A \to \alpha \in R$, can fire since ${}^\bullet t_k = \{\beta^{-1}(A)\}$ and $\mu_k(\beta^{-1}(A)) = |w_k|_A > 0$. If $k = n$, then $\mu_n(p) = 0$ for all $p \in P$ as $w_n \in \Sigma^*$, i.e., $|w_k|_A = 0$ for all $A \in V$.

Let $\nu = t_1 t_2 \cdots t_n$ be an occurrence sequence of transitions of N enabled at ι and finished at μ_n where $\mu_n(p) = 0$ for all $p \in P$. By induction on the number $1 \leq k \leq n$ of occurrence steps we show that $S \xRightarrow{r_1 r_2 \cdots r_n} w, w \in \Sigma^*$, is a derivation in G where $r_1 r_2 \cdots r_n = \gamma(t_1 t_2 \cdots t_n)$.

For $k = 1$ we have $\iota \xrightarrow{t_1} \mu_1$. Then the rule $r_1 = \gamma^{-1}(t_1) : S \to \alpha \in R$ can also be applied and $S \Rightarrow_{r_1} w_1 = \alpha$. By definition, for each $A \in V$, $|w_1|_A = \mu_1(\beta^{-1}(A))$.

We suppose that for $1 \leq m \leq k-1 < n$, $S \xRightarrow{r_1 r_2 \cdots r_m} w_m \in (V \cup \Sigma)^*$ is a derivation in G where $r_1 r_2 \cdots r_m = \gamma(t_1 t_2 \cdots t_m)$. Then for each $A \in V$ and $1 \leq i \leq m$, $|w_i|_A = \mu_i(p)$ where $A = \beta(p)$. If $m = k$, the rule $r_k : A \to \alpha \in R$, $r_k = \gamma(t_k)$, can be applied since $|w_k|_A > 0$. For $k = n$, $\mu_n(p) = 0$ for all $p \in P$ and consequently, $|w_n|_A = \mu_n(\beta^{-1}(A)) = 0$ for all $A \in V$, i.e., $w_n \in \Sigma^*$. □

3 Petri Net Controlled Grammars and Examples

Now we define a *k-Petri net*, i.e., a cf Petri net with additional k places and additional arcs from/to these places to/from transitions of the net, the pre-sets and post-sets of the additional places are disjoint.

Definition 2. *Let $G = (V, \Sigma, S, R)$ be a context-free grammar with its corresponding cf Petri net $N = (P, T, F, \phi, \beta, \gamma, \iota)$. Let k be a positive integer and let $Q = \{q_1, q_2, \ldots, q_k\}$ be a set of new places called counters. A k-Petri net is a construct $N_k = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ where*

- $E = \{(t, q_i) \mid t \in T_1^i, 1 \leq i \leq k\} \cup \{(q_i, t) \mid t \in T_2^i, 1 \leq i \leq k\}$ *such that $T_1^i \subset T$ and $T_2^i \subset T$, $1 \leq i \leq k$ where $T_l^i \cap T_l^j = \emptyset$ for $1 \leq l \leq 2$, $T_1^i \cap T_2^j = \emptyset$ for $1 \leq i < j \leq k$ and $T_1^i = \emptyset$ if and only if $T_2^i = \emptyset$ for any $1 \leq i \leq k$.*
- *the weight function $\varphi(x, y)$ is defined by $\varphi(x, y) = \phi(x, y)$ if $(x, y) \in F$ and $\varphi(x, y) = 1$ if $(x, y) \in E$,*
- *the labeling function $\zeta : (P \cup Q) \to V \cup \{\lambda\}$ is defined by $\zeta(p) = \beta(p)$ if $p \in P$ and $\zeta(p) = \lambda$ if $p \in Q$,*
- *the initial marking μ_0 is defined by $\mu_0(\beta^{-1}(S)) = 1$ and $\mu_0(p) = 0$ for all $p \in (P \cup Q) - \{\beta^{-1}(S)\}$,*
- *τ is the final marking where $\tau(p) = 0$ for all $p \in (P \cup Q)$.*

Definition 3. *A k-Petri net controlled grammar (in short, a k-PN controlled grammar) is a quintuple $G = (V, \Sigma, S, R, N_k)$ where V, Σ, S, R are defined as for a context-free grammar and N_k is a k-Petri net with respect to the context-free grammar (V, Σ, S, R).*

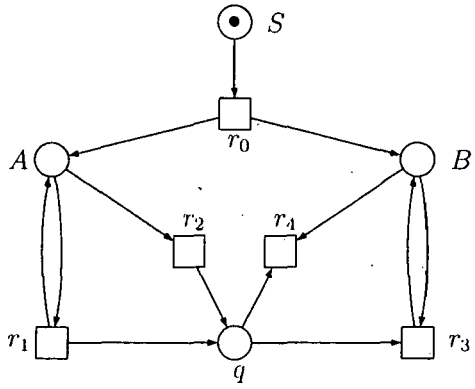

Figure 2: A 1-Petri net N_1

Definition 4. *The language generated by a k-Petri net controlled grammar G consists of all strings $w \in \Sigma^*$ such that there is a derivation*

$$S \overset{r_1 r_2 \cdots r_n}{\Longrightarrow} w \text{ where } t_1 t_2 \cdots t_n = \gamma^{-1}(r_1 r_2 \cdots r_n) \in T^*$$

is an occurrence sequence of the transitions of N_k enabled at the initial marking μ_0 and finished at the final marking τ.

We denote the family of languages generated by k-PN controlled grammars (with erasing rules) by $\mathbf{PN}_k$ ($\mathbf{PN}_k^{\lambda}$), $k \geq 1$. We also use bracket notation $\mathbf{PN}_k^{[\lambda]}$ in order to say that a statement holds in both cases: with and without erasing rules.

We give two examples which will be used in the sequel.

Example 2. Figure 2 illustrates a 1-Petri net N_1 which is constructed from the cf Petri net N in Figure 1 adding a single counter place q. Let $G_2 = (V, \Sigma, S, R, N_1)$ be the 1-PN controlled grammar where V, Σ, S, R are defined as for the grammar G_1 in Example 1. It is not difficult to see that $L(G_2) = \{a^n b^n c^n \mid n \geq 1\}$.

Example 3. Let G_3 be a 2-PN controlled grammar with the production rules:

$$\begin{array}{lll} r_0 : S \to A_1 B_1 A_2 B_2, & r_1 : A_1 \to a_1 A_1 b_1, & r_2 : A_1 \to a_1 b_1, \\ r_3 : B_1 \to c_1 B_1, & r_4 : B_1 \to c_1, & r_5 : A_2 \to a_2 A_2 b_2, \\ r_6 : A_2 \to a_2 b_2, & r_7 : B_2 \to c_2 B_2, & r_8 : B_2 \to c_2 \end{array}$$

and the corresponding 2-Petri net N_2 is given in Figure 3. Then it is easy to see that G_3 generates the language $L(G_3) = \{a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m \mid n, m \geq 1\}$.

Lemma 1. *The language $L' = \{a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m \mid n, m \geq 1\}$ cannot be generated by a 1-PN controlled grammar.*

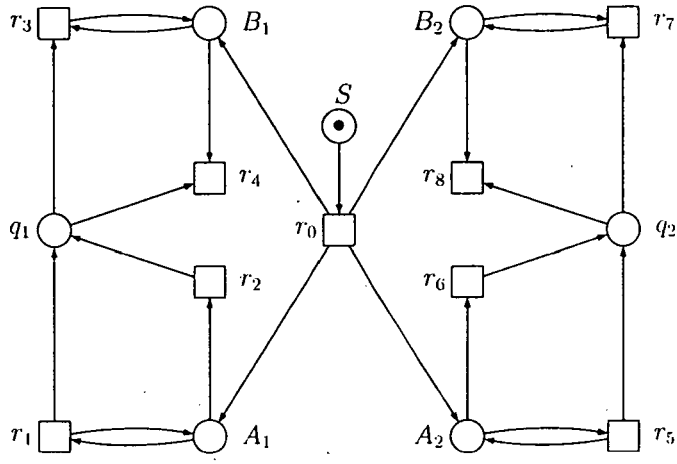

Figure 3: A 2-Petri net N_2

Proof. Suppose the contrary: there is a 1-Petri net controlled grammar $G = (V, \Sigma, S, R, N_1)$ where $\Sigma = \{a_1, b_1, c_1, a_2, b_2, c_2\}$ such that $L(G) = L'$. Let $w = a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m$. Since the set V is finite, and if n and m are chosen sufficiently large, every derivation $S \Rightarrow^* w$ in G contains a subderivation of the form D: $A \Rightarrow^* xAy$ where $A \in V$ and $x, y \in \Sigma^*$ with $xy \neq \lambda$. As L' is infinite, there are words with enough large length obtained by iterating such a derivation D arbitrarily many times. Suppose

$$S \Rightarrow^* uAv \Rightarrow^* uxAyv \Rightarrow^* \cdots \Rightarrow^* ux^n Ay^n v \Rightarrow^* w' \in \Sigma^* \tag{1}$$

is also a derivation in G. Then x^n and y^n are substrings of w'. By the structure of the words of L', x and y can be only powers of two symbols from $\Sigma \cup \{\lambda\}$. Therefore, in order to generate a word $w = a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m \in L'$ for large n and m, we need at least three subderivations of the form

$$D_1 : A_1 \Rightarrow^* x_1 A_1 y_1, \tag{2}$$

$$D_2 : A_2 \Rightarrow^* x_2 A_2 y_2, \tag{3}$$

$$D_3 : A_3 \Rightarrow^* x_3 A_3 y_3 \tag{4}$$

where $x_1, x_2, x_3, y_1, y_2, y_3$ are powers of the symbols from Σ, i.e.,

$$x_i = \alpha_i^{k_i} \text{ and } y_i = \beta_i^{l_i} \text{ where } \alpha_i, \beta_i \in \Sigma \text{ and } k_i + l_i \geq 1, i = 1, 2, 3.$$

First, we assume that (1) has exactly three subderivations of the form (2)–(4). According to the production and consumption of tokens by the subderivations (2)–(4) the following cases can occur:

Case 1. One of the derivations (2)–(4) does not produce and consume any token. Without loss of generality we can assume that this derivation is (2). If

$$S \Rightarrow^* uA_1v \Rightarrow^* uwv \in L'$$

then for any $k > 1$ we apply (2) k times and get a string which is not in L', i.e.

$$S \Rightarrow^* uA_1v \Rightarrow^* ux_1A_1y_1v \Rightarrow^* ux_1^2A_1y_1^2v \Rightarrow^* ux_1^kA_1y_1^kv \Rightarrow^* ux_1^kwy_1^kv \notin L'$$

since (2) increases only the powers of at most two letters.

Case 2. One of the subderivations (2)–(4) produces tokens and another one consumes tokens. Without loss of generality we assume that (2) produces $p \geq 1$ tokens and (3) consumes $q \geq 1$ tokens.

Suppose

$$S \Rightarrow^* u_1A_1u_2A_2u_3 \Rightarrow^* u_1w_1u_2w_2u_3 \in L'.$$

Then the derivation

$$\begin{aligned} S &\Rightarrow^* u_1A_1u_2A_2u_3 \\ &\Rightarrow^* u_1x_1A_1y_1u_2A_2u_3 \Rightarrow^* u_1x_1^kA_1y_1^ku_2A_2u_3 \\ &\Rightarrow^* u_1x_1^kA_1y_1^ku_2x_2A_2y_2u_3 \Rightarrow^* u_1x_1^kA_1y_1^ku_2x_2^lA_2y_2^lu_3 \\ &\Rightarrow^* u_1x_1^kw_1y_1^ku_2x_2^lw_2y_2^lu_3 \end{aligned}$$

where $k, l \geq 1$, is also in G. It can be done by choosing the numbers k, l in such a way, that $kp - lq = 0$, thus we can choose k and l as $k = q$ and $l = p$ and still get a string $w' \in L'$. Now

- if $1 \leq |\{\alpha_1, \beta_1, \alpha_2, \beta_2\} \cap \{a_i, b_i, c_i\}| \leq 2$, $i = 1$ or $i = 2$ then $w' \notin L'$ as the powers of at most two symbols are increased;
- if $\{\alpha_1, \beta_1, \alpha_2, \beta_2\} \cap \{a_i, b_i, c_i\} \neq \emptyset$ for both $i = 1$ and $i = 2$ then $1 \leq |\{\alpha_1, \beta_1, \alpha_2, \beta_2\} \cap \{a_i, b_i, c_i\}| \leq 2$ for $i = 1$ or $i = 2$ and again $w' \notin L'$.

From the above it follows that $\{\alpha_1, \beta_1, \alpha_2, \beta_2\} = \{a_i, b_i, c_i, \lambda\}$ for $i = 1$ or $i = 2$. Without loss of generality we assume that $i = 1$. But from the subderivation (4) (which produces or consumes tokens) it follows that $\alpha_3, \beta_3 \notin \{a_1, b_1, c_1\}$ and at least one of them belongs to $\{a_2, b_2, c_2\}$. Again we get the contradiction since (4) can increase the powers of at most two symbols from $\{a_2, b_2, c_2\}$.

If the derivation has the form

$$S \Rightarrow^* u_1A_1u_4 \Rightarrow^* u_1u_2A_2u_3u_4 \Rightarrow^* u_1u_2wu_3u_4,$$

then one gets that $\{x_1, y_1, x_2, y_2\}$ contains only two elements from Σ and a contradiction follows as above.

Case 3. Two of the subderivations of (2)-(4) produce (consume) tokens and the other consumes (produces). Without loss of generality we assume that (2) and (3) produces p_1 and p_2 tokens, respectively and (4) consumes q tokens. If

$$S \Rightarrow^* u_1 A_1 u_2 A_2 u_3 A_3 u_4 \Rightarrow^* u_1 w_1 u_2 w_2 u_3 w_3 u_4 \in L',$$

then the derivation

$$\begin{aligned} S &\Rightarrow^* u_1 A_1 u_2 A_2 u_3 A_3 u_4 \\ &\Rightarrow^* u_1 x_1 A_1 y_1 u_2 x_2 A_2 y_2 u_3 x_3 A_3 y_3 u_4 \\ &\Rightarrow^* u_1 x_1^{k_1} A_1 y_1^{k_1} u_2 x_2^{k_2} A_2 y_2^{k_2} u_3 x_3^{l} A_3 y_3^{l} u_4 \\ &\Rightarrow^* u_1 x_1^{k_1} w_1 y_1^{k_1} u_2 x_2^{k_2} w_2 y_2^{k_2} u_3 x_3^{l} w_3 y_3^{l} u_4 = w' \end{aligned} \tag{5}$$

is also in G. By the definition of the final marking, we have $k_1 p_1 + k_2 p_2 - lq = 0$. For instance, if we choose k_1, k_2, l as $k_1 = p_2 q$, $k_2 = p_1 q$ and $l = 2p_1 p_2$, this equality holds. By structure of a derivation there are two possibilities:

$$\{\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3\} = \{a_1, b_1, c_1, a_2, b_2, c_2, \lambda\} \tag{6}$$

or

$$\{\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3\} = \{a_i, b_i, c_i, \lambda\} \text{ where } i = 1 \text{ or } i = 2. \tag{7}$$

Consider (6), here we only have the case $\alpha_1 = a_1$, $\beta_1 = b_1$, $\alpha_2 = c_1$, $\beta_2 = a_2$, $\alpha_3 = b_2$ and $\beta_3 = c_2$. It follows that the powers of all symbols of w' are the same. But from (5), by continuing the derivation, we get a string which is not in L':

$$\begin{aligned} S &\Rightarrow^* u_1 x_1^{k_1} A_1 y_1^{k_1} u_2 x_2^{k_2} A_2 y_2^{k_2} u_3 x_3^{l} A_3 y_3^{l} u_4 \\ &\Rightarrow^* u_1 x_1^{k_1} w_1 y_1^{k_1} u_2 x_2^{k_2} w_2 y_2^{k_2} u_3 x_3^{l} A_3 y_3^{l} u_4 \\ &\Rightarrow^* u_1 x_1^{k_1} w_1 y_1^{k_1} u_2 x_2^{k_2} w_2 y_2^{k_2} u_3 x_3^{2l} A_3 y_3^{2l} u_4 \\ &\Rightarrow^* u_1 x_1^{k_1} w_1 y_1^{k_1} u_2 x_2^{2k_2} w_2 y_2^{2k_2} u_3 x_3^{3l} w_3 y_3^{3l} u_4 \notin L' \end{aligned}$$

where the powers of four symbols are increased.

Now consider (7). Let $i = 1$. From Case 2, we can conclude that one of the following three cases is possible:

$$\begin{array}{lllll} (a) & \{\alpha_1, \beta_1\} = \{a_1, b_1\}, & \{\alpha_2, \beta_2\} = \{\lambda\}, & \{\alpha_3, \beta_3\} = \{c_1, \lambda\}, \\ (b) & \{\alpha_1, \beta_1\} = \{\lambda\}, & \{\alpha_2, \beta_2\} = \{a_1, b_1\}, & \{\alpha_3, \beta_3\} = \{c_1, \lambda\}, \\ (c) & \{\alpha_1, \beta_1\} = \{a_1, \lambda\}, & \{\alpha_2, \beta_2\} = \{b_1, \lambda\}, & \{\alpha_3, \beta_3\} = \{c_1, \lambda\}. \end{array}$$

Cases (a) and (b) are similar to *Case 2.* If we choose $k_1 = 3p_2 l$, $k_2 = 2p_1 l$ and $q = 5p_1 p_2$ in case (c), we again get different powers for symbols a_1, b_1, c_1, i.e., $w' \notin L'$.

Next, we analyze the general case: let the derivation (1) have $n \geq 4$ subderivations of the form $D_i : A_i \to x_i A_i y_i$ where $A_i \in V$, $x_i = \alpha_i^{l_i}$ and $y_i = \beta_i^{l_i}$, $\alpha_i, \beta_i \in \Sigma$,

$l_i + l_i' \geq 1$, $1 \leq i \leq n$. Without loss of generality we can assume that for some $1 \leq s \leq n-1$, the derivations D_i, $1 \leq i \leq s$, produce p_i tokens and the derivations D_j, $s+1 \leq j \leq n$, consume q_j tokens. If

$$S \Rightarrow^* u_1 A_1 u_2 A_2 u_3 \cdots u_n A_n u_{n+1} \Rightarrow^* u_1 w_1 u_2 w_2 u_3 \cdots u_n w_n u_{n+1} = w \in L', \quad (8)$$

then by assumption,

$$\begin{aligned} S &\Rightarrow^* u_1 A_1 u_2 A_2 u_3 \cdots u_n A_n u_{n+1} \\ &\Rightarrow^* u_1 x_1 A_1 y_1 u_2 x_2 A_2 y_2 u_3 \cdots u_n x_n A_n y_n u_{n+1} \\ &\Rightarrow^* u_1 x_1^{k_1} A_1 y_1^{k_1} u_2 x_2^{k_2} A_2 y_2^{k_2} u_3 \cdots u_n x_n^{k_n} A_n y_n^{k_n} u_{n+1} \\ &\Rightarrow^* u_1 x_1^{k_1} w_1 y_1^{k_1} u_2 x_2^{k_2} w_2 y_2^{k_2} u_3 \cdots u_n x_n^{k_n} w_n y_n^{k_n} u_{n+1} = w' \in L'. \end{aligned} \quad (9)$$

According to the definition of the final marking, we have

$$\sum_{i=1}^{s} k_i p_i - \sum_{i=s+1}^{n} k_i q_i = 0.$$

and

$$\{\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_n, \beta_n\} = \{a_1, b_1, c_1, a_2, b_2, c_2, \lambda\}.$$

If for some $1 \leq i \leq n$, $\alpha_i = c_1$ and $\beta_i = a_2$, then all symbols in w' have the same power. Then by continuing two subderivations one of which produces tokens and the other consumes, one increases the powers of at most four symbols, and get a string $w'' \notin L'$.

Let, for some $2 \leq i \leq n-2$,

$$\{\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_i, \beta_i\} = \{a_1, b_1, c_1, \lambda\} \quad (10)$$

and

$$\{\alpha_{i+1}, \beta_{i+1}, \alpha_{i+2}, \beta_{i+2}, \ldots, \alpha_n, \beta_n\} = \{a_2, b_2, c_2, \lambda\}. \quad (11)$$

It follows that at least one of the subderivations which generate symbols in (10) (symbols in (11)) produces and another subderivation consumes tokens, since symbols a_i, b_i, c_i, $i = 1, 2$, have the same power. Then the tokens produced by a subderivation D_j, for some $1 \leq j \leq i$, can be consumed by a subderivation D_k, for some $i+1 \leq k \leq n$ as the both group of subderivations use the same counter, which result that the powers of at most two symbols from a_1, b_1, c_1 and a_2, b_2, c_2 are increased, i.e., a string $w' \notin L'$ is generated. In all cases, we get contradiction to our assumption $L' = L(G)$. □

4 Hierarchy Results

We start with a simple fact.

Lemma 2. $\mathbf{CF} \subsetneq \mathbf{PN}_1$.

Proof. It is clear that $\mathbf{CF} \subseteq \mathbf{PN}_1$ if we take $T_1 = T_2 = \emptyset$. From Example 2 it follows that $\mathbf{CF} \subsetneq \mathbf{PN}_1$. □

Now we present some relations to (positive) additive valence languages.

Lemma 3. $\mathbf{PN}_1^{[\lambda]} \subseteq \mathbf{pV}^{[\lambda]}$.

Proof. Let $G = (V, \Sigma, S, R, N_1)$ be a 1-PN controlled grammar (with or without erasing rules) where $N_1 = (P \cup \{q\}, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ is a corresponding 1-Petri net with the counter q (with the notions of Definition 2). We define a positive valence grammar $G' = (V, \Sigma, S, R, v)$ where V, Σ, S, R are defined as for the grammar G and for each $r \in R$, the mapping v is defined by

$$v(r) = \begin{cases} 1 & \text{if } \gamma^{-1}(r) \in {}^\bullet q, \\ -1 & \text{if } \gamma^{-1}(r) \in q^\bullet, \\ 0 & \text{otherwise.} \end{cases}$$

Let $S \overset{\pi}{\Rightarrow} w, w \in \Sigma^*$, $\pi = r_1 r_2 \cdots r_k$, be a derivation in G. Then $\nu = t_1 t_2 \cdots t_k = \gamma^{-1}(r_1 r_2 \cdots r_k)$ is an occurrence sequence of transitions of N_1 enabled at the initial marking μ_0 and finished at the final marking τ, i.e.,

$$\mu_0 \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_k} \mu_k = \tau$$

By definition, if $|\nu|_t > 0$ for some $t \in {}^\bullet q$ then there is a transition $t' \in q^\bullet$ such that $|\nu|_{t'} > 0$. Let

$$U_1 = \{t_{1,1}, t_{1,2}, \ldots, t_{1,k_1}\} \subseteq {}^\bullet q \text{ where } |\nu|_{t_{1,j}} > 0, 1 \leq j \leq k_1$$

and

$$U_2 = \{t_{2,1}, t_{2,2}, \ldots, t_{2,k_2}\} \subseteq q^\bullet \text{ where } |\nu|_{t_{2,j}} > 0, 1 \leq j \leq k_2.$$

Since $\mu_i(q) \geq 0$ for each occurrence step $1 \leq i \leq k$, we have $|\nu|_{U_1} \geq |\nu|_{U_2}$, consequently, $v(r_1) + v(r_2) + \ldots + v(r_j) \geq 0$ for any $1 \leq j < k$ and from $\mu_0(q) = \tau(q) = 0$, $\tau \in M$, it follows that

$$\sum_{t \in U_1} |\nu|_t - \sum_{t \in U_2} |\nu|_t \overset{\text{def}}{=} \sum_{i=1}^{k} v(r_i) = 0.$$

Hence, $L(G) \subseteq L(G')$.

Let $D : S \overset{r_1 r_2 \cdots r_k}{\Longrightarrow} w \in \Sigma^*$ be a derivation in G' where $v(r_1) + v(r_2) + \ldots + v(r_k) = 0$ and $v(r_1) + v(r_2) + \ldots + v(r_j) \geq 0$ for any $1 \leq j < k$. By construction of G', D is also a derivation in (V, Σ, S, R).

According to the bijection $\gamma : T \to R$, there is an occurrence sequence $\nu = t_1 t_2 \cdots t_k$, $\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_k} \mu_k$, in N_1 such that $\nu = \gamma^{-1}(r_1 r_2 \cdots r_k)$.

$\mu = \mu_0$ since D starts from S, i.e., $\mu_0(\beta^{-1}(S)) = 1$ and $\mu_0(\beta^{-1}(x)) = 0$ for all $x \in (V \cup \Sigma) - \{S\}$ as well as $\mu_0(q) = 0$.

Since $w \in \Sigma^*$, we have $\mu_k(\beta^{-1}(x)) = 0$ for all $x \in V$. From $\sum_{i=1}^{j} v(r_i) \geq 0$, it follows that $\mu_j(q) \geq 0$ for any $1 \leq j < k$.

$$\sum_{i=1}^{k} v(r_i) \stackrel{\text{def}}{=} \sum_{\gamma^{-1}(r) \in {}^\bullet q} v(r) + \sum_{\gamma^{-1}(r) \in q^\bullet} v(r) = 0$$

shows that $\mu_k(q) = 0$. Therefore $\mu_k = \tau$. Consequently, $L(G') \subseteq L(G)$. □

Lemma 4. $\mathbf{aV}^{[\lambda]} \subsetneq \mathbf{PN}_2^{[\lambda]}$.

Proof. Let $G = (V, \Sigma, S, R, v)$ be an additive valence grammar (with or without erasing rules). Without loss of generality we can assume that $v(r) \in \{1, 0, -1\}$ for each $r \in R$ (Lemma 2.1.10 in [2]).

For each rule $r : A \to \alpha \in R$, $v(r) \neq 0$ we add a nonterminal symbol A_r and a pair of rules $r' : A \to A_r$, $r'' : A_r \to \alpha$ and we set

$$\begin{aligned} V' =& V \cup \{A_r \mid r : A \to \alpha \in R, v(r) \neq 0\}, \\ R' =& R \cup \{r' : A \to A_r, r'' : A_r \to \alpha \mid r : A \to \alpha \in R, v(r) \neq 0\}. \end{aligned}$$

Let $N = (P, T, F, \phi, \beta, \gamma, \iota)$ be a cf Petri net with respect to the context-free grammar (V', Σ, S, R'). We construct a 2-Petri net $N_2 = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ where $Q = \{q, q'\}$ and $E = F_1 \cup F_2$ with

$$\begin{aligned} F_1 =& \{(t, q) \mid t = \gamma^{-1}(r), r \in R \text{ and } v(r) = 1\} \\ & \cup \{(t', q') \mid t' = \gamma^{-1}(r'), r \in R \text{ and } v(r) = -1\}, \\ F_2 =& \{(q, t) \mid t = \gamma^{-1}(r), r \in R \text{ and } v(r) = -1\} \\ & \cup \{(q', t') \mid t' = \gamma^{-1}(r'), r \in R \text{ and } v(r) = 1\}. \end{aligned}$$

The rest components of N_2 are defined the same as those in the definition. Consider the 2-PN controlled grammar $G' = (V', \Sigma, S, R', N_2)$.

Let $D : S \stackrel{\pi}{\Rightarrow} w, w \in \Sigma^*$, $\pi = r_1 r_2 \cdots r_n$, be a derivation in G'. Then $\sigma = t_1 t_2 \cdots t_n = \gamma^{-1}(r_1 r_2 \cdots r_n)$ is an occurrence sequence enabled at the initial marking μ_0 and finished at the final marking τ. By construction,

$$\sum_{i=1}^{n} v(r_i) = \sum_{t \in {}^\bullet q} |\sigma|_t + \sum_{t \in q'^\bullet} |\sigma|_t - \sum_{t \in q^\bullet} |\sigma|_t - \sum_{t \in {}^\bullet q'} |\sigma|_t = 0$$

since

$$\sum_{t \in {}^\bullet q} |\sigma|_t = \sum_{t \in q^\bullet} |\sigma|_t = \sum_{i=1}^{n} \mu_i(q) \text{ and } \sum_{t \in {}^\bullet q'} |\sigma|_t = \sum_{t \in q'^\bullet} |\sigma|_t = \sum_{i=1}^{n} \mu_i(q').$$

It follows that D is also a derivation in G.

Let $D' : S \stackrel{r_1 r_2 \cdots r_n}{\Longrightarrow} w, w \in \Sigma^*$ be a derivation in G. For each $1 \leq k \leq n$,

(1) if $\sum_{i=1}^{k} v(r_i) > 0$, then for the rule r_k with $v(r_k) \in \{1, 0, -1\}$ in G choose the rule r_k in G';

(2) if $\sum_{i=1}^{k} v(r_i) < 0$, then for the rule r_k with $v(r_k) \neq 0$ in G choose the rules r'_k and r''_k in G'; if $v(r_k) = 0$ then choose r_k in G'.

(3) if $\sum_{i=1}^{k} v(r_i) = 0$, then for the rule r_k with $v(r_k) \in \{-1, 0\}$ in G choose the rule r_k in G'; if $v(r_k) = 1$, then choose r'_k, r''_k in G'.

Therefore D' is also a derivation in G'. The strict inclusion follows from the fact that

$$\{a_1^n b_1^n c_1^n a_2^m b_2^m c_2^m \mid n, m \geq 1\} \in \mathbf{PN}_2$$

cannot be generated by an additive valence grammar (Example 2.1.7 in [2]). □

The following lemma shows that, for any $n \geq 1$, an n-PN controlled grammar generates a vector language.

Lemma 5. *For $n \geq 1$, $\mathbf{PN}_n^{[\lambda]} \subseteq \mathbf{V}^{[\lambda]}$.*

Proof. Let $G = (V, \Sigma, S, R, N_n)$ be an n-PN controlled grammar (with or without erasing rules) with the n-Petri net $N_n = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$. Let $Q = \{q_1, q_2, \ldots, q_n\}$ and

$$^\bullet q_k = \{t_{k,1,1}, t_{k,1,2}, \ldots, t_{k,1,s(k)}\}$$

where $t_{k,1,i} = \gamma^{-1}(r_{k,1,i})$, $r_{k,1,i} : A_{k,1,i} \to w_{k,1,i}$, $1 \leq k \leq n$, $1 \leq i \leq s(k)$, and

$$q_k^\bullet = \{t_{k,2,1}, t_{k,2,2}, \ldots, t_{k,2,l(k)}\}$$

where $t_{k,2,j} = \gamma^{-1}(r_{k,2,j})$, $r_{k,2,j} : A_{k,2,j} \to w_{k,2,j}$, $1 \leq k \leq n$, $1 \leq j \leq l(k)$.

Let

$$\beta(p_{k,1,i}) = A_{k,1,i},\ 1 \leq k \leq n,\ 1 \leq i \leq s(k)$$

and

$$\beta(p_{k,2,j}) = A_{k,2,j},\ 1 \leq k \leq n,\ 1 \leq j \leq l(k).$$

First, we construct a PN controlled grammar $G' = (V', \Sigma, S, R', N')$ in such a way that each counter place of N' has a single input transition and a single output transition, and we show that the grammars G and G' generate the same language. We set $V' = V \cup \{B_{k,i,j}, C_{k,j,i} \mid 1 \leq k \leq n, 1 \leq i \leq s(k), 1 \leq j \leq l(k)\}$ where $B_{k,i,j}$ and $C_{k,j,i}$, $1 \leq k \leq n$, $1 \leq i \leq s(k)$, $1 \leq j \leq l(k)$, are new nonterminals. R' consists of the following rules

$$\begin{aligned}
R' = (R - \{&r_{k,1,i}, r_{k,2,j} \mid 1 \leq k \leq n,\ 1 \leq i \leq s(k),\ 1 \leq j \leq l(k)\}) \\
&\cup \{r'_{k,1,i,j} : A_{k,1,i} \to B_{k,i,j} \mid 1 \leq k \leq n, 1 \leq i \leq s(k), 1 \leq j \leq l(k)\} \\
&\cup \{r''_{k,1,i,j} : B_{k,i,j} \to w_{k,1,i} \mid 1 \leq k \leq n, 1 \leq i \leq s(k), 1 \leq j \leq l(k)\} \\
&\cup \{r'_{k,2,j,i} : A_{k,2,j} \to C_{k,j,i} \mid 1 \leq k \leq n, 1 \leq i \leq s(k), 1 \leq j \leq l(k)\} \\
&\cup \{r''_{k,2,j,i} : C_{k,j,i} \to w_{k,2,j} \mid 1 \leq k \leq n, 1 \leq i \leq s(k), 1 \leq j \leq l(k)\}
\end{aligned}$$

and $N' = (P' \cup Q', T', F', \varphi', \zeta', \gamma', \mu_0', \tau')$ where the sets of places, transitions and arcs

$$\begin{aligned}
P' =& P \cup \{p_{k,1,i,j} \mid 1 \le k \le n,\ 1 \le i \le s(k),\ 1 \le j \le l(k)\} \\
& \cup \{p_{k,2,j,i} \mid 1 \le k \le n,\ 1 \le i \le s(k),\ 1 \le j \le l(k)\}, \\
Q' =& \{q_{k,i,j} \mid 1 \le k \le n,\ 1 \le i \le s(k), 1 \le j \le l(k)\}, \\
T' =& (T - \bigcup_{k=1}^{n} ({}^\bullet q_k \cup q_k^\bullet)) \\
& \cup \{t'_{k,1,i,j}, t''_{k,1,i,j} \mid 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k)\} \\
& \cup \{t'_{k,2,j,i}, t''_{k,2,j,i} \mid 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k)\},
\end{aligned}$$

$$\begin{aligned}
F' = (F \cup E - \bigcup_{k=1}^{n} & (\{(p_{k,1,i}, t_{k,1,i}), (t_{k,1,i}, q_k) \mid 1 \le i \le s(k)\} \\
& \cup \{(t_{k,1,i}, p) \mid p = \zeta^{-1}(x),\ |w_{k,1,i}|_x > 0, 1 \le i \le s(k)\} \\
& \cup \{(q_k, t_{k,2,j}), (p_{k,2,j}, t_{k,2,j}) \mid 1 \le j \le l(k)\} \\
& \cup \{(t_{k,2,j}, p) \mid p = \zeta^{-1}(x),\ |w_{k,2,j}|_x > 0, 1 \le j \le l(k)\})) \\
\cup \bigcup_{k=1}^{n} \bigcup_{i=1}^{s(k)} \bigcup_{j=1}^{l(k)} & (\{(p_{k,1,i}, t'_{k,1,i,j}), (t'_{k,1,i,j}, p_{k,1,i,j}), (p_{k,1,i,j}, t''_{k,1,i,j}), \\
& (t''_{k,1,i,j}, q_{k,i,j})\} \cup \{(t''_{k,1,i,j}, p) \mid p = \zeta^{-1}(x),\ |w_{k,1,i}|_x > 0\}) \\
\cup \bigcup_{k=1}^{n} \bigcup_{j=1}^{l(k)} \bigcup_{i=1}^{s(k)} & (\{(p_{k,2,j}, t'_{k,2,j,i}), (t'_{k,1,j,i}, p_{k,2,j,i}), (p_{k,2,j,i}, t''_{k,2,j,i}), \\
& (t''_{k,2,j,i}, q_{k,i,j})\} \cup \{(t''_{k,2,j,i}, p) \mid p = \zeta^{-1}(x),\ |w_{k,2,j}|_x > 0\}).
\end{aligned}$$

- The weight function is defined by

$$\varphi'(x,y) = \begin{cases}
\varphi(x,y) & \text{if } (x,y) \in F, \\
\varphi(t_{k,1,i},p) & \text{if } x = t_{k,1,i,j}, y = p = \zeta^{-1}(x), |w_{k,1,i}|_x > 0, \\
& \quad 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k), \\
\varphi(t_{k,2,j},p) & \text{if } x = t_{k,2,j,i}, y = p = \zeta^{-1}(x), |w_{k,2,j}|_x > 0, \\
& \quad 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k), \\
1 & \text{otherwise.}
\end{cases}$$

- The labeling functions are defined by

$$\zeta'(p) = \begin{cases}
\zeta(p) & \text{if } p \in P, \\
B_{k,i,j} & \text{if } p = p_{k,1,i,j}, 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k), \\
C_{k,j,i} & \text{if } p = p_{k,2,j,i}, 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k), \\
\lambda, & \text{if } p = q_{k,i,j}, 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k)
\end{cases}$$

and

$$\gamma'(t) = \begin{cases} \gamma(t) & \text{if } t \in T, \\ r'_{k,1,i,j} & \text{if } t = t'_{k,1,i,j}, 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k), \\ r''_{k,1,i,j} & \text{if } t = t''_{k,1,i,j}, 1 \le k \le n, 1 \le i \le s(k), 1 \le j \le l(k), \\ r'_{k,2,j,i} & \text{if } t = t'_{k,2,j,i}, 1 \le k \le n, 1 \le j \le l(k), 1 \le i \le s(k), \\ r''_{k,2,j,i} & \text{if } t = t''_{k,2,j,i}, 1 \le k \le n, 1 \le j \le l(k), 1 \le i \le s(k). \end{cases}$$

- The initial marking is defined by $\mu'_0(\zeta^{-1}(S)) = 1$ and $\mu'_0(p) = 0$ for all $p \in P' \cup Q' - \{\zeta^{-1}(S)\}$.
- The final marking is defined by $\tau'(p) = 0$ for all $p \in P' \cup Q'$.

By the construction of N', an occurrence sequence of the form

$$\mu_1 \xrightarrow{t'_{k,1,i,j}} \mu_2 \xrightarrow{\sigma'} \mu_3 \xrightarrow{t''_{k,1,i,j}} \mu_4 \xrightarrow{\sigma''} \mu_5 \xrightarrow{t''_{k,2,j,i}} \mu_6 \xrightarrow{\sigma'''} \mu_7 \xrightarrow{t'_{k,2,j,i}} \mu_8 \tag{12}$$

where $\sigma', \sigma'', \sigma''' \in T'^*$ can be replaced by

$$\mu_1 \xrightarrow{t'_{k,1,i,j}} \mu_2 \xrightarrow{t''_{k,1,i,j} \cdot \sigma'} \mu_4 \xrightarrow{\sigma''} \mu_5 \xrightarrow{\sigma''' \cdot t''_{k,2,j,i}} \mu_7 \xrightarrow{t'_{k,2,j,i}} \mu_8. \tag{13}$$

Then, it is clear that (13) can be replaced in N_n by

$$\mu_1 \xrightarrow{t_{k,1,i}} \mu' \xrightarrow{\sigma' \cdot \sigma'' \cdot \sigma'''} \mu'' \xrightarrow{t_{k,2,j}} \mu_8.$$

Conversely, an occurrence sequence of the form

$$\mu_1 \xrightarrow{t_{k,1,i}} \mu_2 \xrightarrow{\sigma} \mu_3 \xrightarrow{t_{k,2,j}} \mu_4$$

in N_n can be replaced in N' by

$$\mu_1 \xrightarrow{t'_{k,1,i,j}} \mu' \xrightarrow{t''_{k,1,i,j}} \mu_2 \xrightarrow{\sigma} \mu_3 \xrightarrow{t'_{k,2,j,i}} \mu'' \xrightarrow{t''_{k,2,j,i}} \mu_4.$$

Correspondingly, without loss of generality we can change the order of the application of rules of derivations in the grammars G and G'. Therefore, $L(G) = L(G')$.

Now we show that the grammar G' generates a vector language. By the construction of N', $|{}^\bullet q| = |q^\bullet| = 1$ for all $q \in Q'$.

We associate with each pair of rules $r_1, r_2 \in R'$ where $r_1 = \gamma'(t_1)$, $t_1 \in {}^\bullet q$ and $r_2 = \gamma'(t_2)$, $t_2 \in q^\bullet$, $q \in Q'$, the matrix $m = (r_1, r_2)$ and with each rule $r \in R' - \{r' = \gamma'(t') \mid t' \in {}^\bullet Q' \cup Q'^\bullet\}$, the matrix $m = (r)$. We consider a vector grammar $G'' = (V', \Sigma, S, M)$ where M is the set of all matrices constructed above.

Let $S \overset{\pi}{\Rightarrow} w, w \in \Sigma^*$, $\pi = r_1 r_2 \cdots r_n$, is a derivation in G' where $\iota \xrightarrow{\nu} \tau$ with $\nu = t_1 t_2 \cdots t_n = \gamma'^{-1}(\pi)$.

Let ${}^\bullet q = \{t\}$ and $q^\bullet = \{t'\}$ for some $q \in Q'$. If t in ν, i.e., $|\nu|_t > 0$ then t' is also in ν and $|t_1 t_2 \cdots t_k|_t \ge |t_1 t_2 \cdots t_k|_{t'}$ for each $1 \le k \le n$, moreover, by

the definition of the final marking, $|\nu|_t = |\nu|_{t'}$. By the bijection γ', $m = (r, r')$, $r = \gamma'(t), r' = \gamma'(t')$ is in π and $|r_1 r_2 \cdots r_k|_r \geq |r_1 r_2 \cdots r_k|_{r'}$ for each $1 \leq k \leq n$ as well as $|\pi|_r = |\pi|_{r'}$. Hence, $\pi \in \mathrm{Shuf}^*(M)$.

Let $S \overset{\pi}{\Rightarrow} w, w \in \Sigma^*$, $\pi = r_1 r_2 \cdots r_n \in \mathrm{Shuf}^*(M)$, be a derivation in G'' then again by the bijection γ', $\nu = t_1 t_2 \cdots t_n = \gamma^{-1}(r_1 r_2 \cdots r_n)$ is an occurrence sequence of transitions of N': $\mu_0 \xrightarrow{\nu} \mu_n$. Since the derivation π starts from S (i.e., S is the only symbol at the starting sentential form), $\mu_0(\beta^{-1}(S)) = 1$ and $\mu_0(p) = 0$ for all $p \in P - \{\beta^{-1}(S)\}$. It follows that $\mu_0 = \mu'_0$. On the other hand, from $w \in \Sigma^*$, it follows that $\mu_n(\beta^{-1}(x)) = 0$ for all $x \in V$. From $\pi \in \mathrm{Shuf}^*(M)$, if the rules r, r' of a matrix $m = (r, r')$ in π then $|r_1 r_2 \cdots r_k|_r \geq |r_1 r_2 \cdots r_k|_{r'}$ for each $1 \leq k \leq n$ and $|\pi|_r = |\pi|_{r'}$. By the bijection γ, $|t_1 t_2 \cdots t_k|_t \geq |t_1 t_2 \cdots t_k|_{t'}$ for each $1 \leq k \leq n$ where $t = \gamma^{-1}(r)$, $\gamma^{-1}(r')$ and $|\nu|_t = |\nu|_{t'}$. It follows that $\mu_n(q) = 0$ for all $q \in Q'$. Hence, $\mu_n = \tau'$. □

Theorem 1. *For $k \geq 1$, $\mathbf{PN}_k^{[\lambda]} \subsetneq \mathbf{PN}_{k+1}^{[\lambda]}$.*

Proof. We first prove that $\mathbf{PN}_1^{[\lambda]} \subseteq \mathbf{PN}_2^{[\lambda]}$.

Let $G = (V, \Sigma, S, R, N_1)$ be a 1-PN controlled grammar (with or without erasing rules) where $N_1 = (P \cup \{q\}, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ 1-PN with the counter place q. Let

$$^\bullet q = \{t_{1,1}, t_{1,2}, \ldots, t_{1,k_1}\}, k_1 \geq 1 \text{ and } q^\bullet = \{t_{2,1}, t_{2,2}, \ldots, t_{2,k_2}\}, k_2 \geq 1$$

where $t_{i,j} = \gamma^{-1}(r_{i,j})$, $r_{i,j} : A_{i,j} \to w_{i,j}$, $1 \leq i \leq 2$, $1 \leq j \leq k_i$ and by definition $^\bullet q \cap q^\bullet = \emptyset$. Let $p_{i,j} = \zeta^{-1}(A_{i,j})$, $1 \leq i \leq 2$, $1 \leq j \leq k_i$.

We set

$$V' = V \cup \{B_{i,j} \mid 1 \leq i \leq 2, 1 \leq j \leq k_i\}$$

where $B_{i,j}$, $1 \leq i \leq 2$, $1 \leq j \leq k_i$, are new nonterminal symbols, introduced for each transition $t_{i,j}$.

For each rule $r_{i,j} : A_{i,j} \to w_{i,j}$, $1 \leq i \leq 2$, $1 \leq j \leq k_i$, we add the new rules $r'_{i,j} : A_{i,j} \to B_{i,j}$, $r''_{i,j} : B_{i,j} \to w_{i,j}$. Let R' be the set of all rules of R and all rules constructed above, i.e.,

$$\begin{aligned} R' = R &\cup \{r'_{1,j} : A_{1,j} \to B_{1,j} \mid \gamma^{-1}(A_{1,j} \to w_{1,j}) \in {}^\bullet q, 1 \leq j \leq k_1\} \\ &\cup \{r''_{1,j} : B_{1,j} \to w_{1,j} \mid \gamma^{-1}(A_{1,j} \to w_{1,j}) \in {}^\bullet q, 1 \leq j \leq k_1\} \\ &\cup \{r'_{2,j} : A_{2,j} \to B_{2,j} \mid \gamma^{-1}(A_{2,j} \to w_{2,j}) \in q^\bullet, 1 \leq j \leq k_2\} \\ &\cup \{r''_{2,j} : B_{2,j} \to w_{2,j} \mid \gamma^{-1}(A_{2,j} \to w_{2,j}) \in q^\bullet, 1 \leq j \leq k_2\}. \end{aligned}$$

We construct a 2-PN controlled grammar $G' = (V', \Sigma, S, R', N_2)$ where V' and R' are defined above and $N_2 = (P', T', F', \varphi', \zeta', \gamma', \mu'_0, \tau')$ is constructed as follows:

$$
\begin{aligned}
P' &= P \cup \{p'_{i,j} \mid 1 \le i \le 2,\ 1 \le j \le k_i\} \cup \{q, q'\},\\
T' &= T \cup \{t'_{i,j}, t''_{i,j} \mid 1 \le i \le 2,\ 1 \le j \le k_i\},\\
F' &= F \cup \bigcup_{i=1}^{2} \bigcup_{j=1}^{k_i} (\{(p_{i,j}, t'_{i,j}), (t'_{i,j}, p'_{i,j}), (p'_{i,j}, t''_{i,j})\}\\
&\quad \cup \{(t''_{i,j}, p) \mid p = \zeta^{-1}(x),\ |w_{i,j}|_x > 0\})\\
&\quad \cup \{(t''_{1,j}, q') \mid 1 \le j \le k_1\}\\
&\quad \cup \{(q', t''_{2,j}) \mid 1 \le j \le k_2\}.
\end{aligned}
$$

For the weight function we set

$$
\varphi'(x,y) = \begin{cases} \varphi(x,y) & \text{if } (x,y) \in F,\\ \varphi(t_{i,j},p) & \text{if } x = t''_{i,j}, y = p = \zeta^{-1}(x), |w_{i,j}|_x > 0,\\ & \quad 1 \le i \le 2, 1 \le j \le k_i,\\ 1 & \text{otherwise.} \end{cases}
$$

The initial and final markings are defined by $\mu'_0(\zeta'^{-1}(S)) = 1$, $\mu'_0(p) = 0$ for all $p \in P' - \{\zeta'^{-1}(S)\}$ and $\tau'(p) = 0$ for all $p \in P'$.

The inclusion $L(G) \subseteq L(G')$ is obvious, which directly follows from the construction of G'.

Let $S \overset{\pi}{\Rightarrow} w, w \in \Sigma^*$, $\pi = r_1 r_2 \cdots r_n$, be a derivation in G' with the occurrence sequence $\nu = t_1 t_2 \cdots t_n = \zeta'^{-1}(\pi)$ of transitions of N_2 enabled at the initial marking μ'_0 and finished at the final marking τ'. It is clear that for some $1 \le i \le 2$, $1 \le j \le k_i$, if a rule $r'_{i,j} : A_{i,j} \to B_{i,j}$ in π, i.e., $|\pi|_{r'_{i,j}} > 0$, then the rule $r''_{i,j} : B_{i,j} \to w_{i,j}$ is also in π, i.e., $|\pi|_{r''_{i,j}} > 0$, moreover, $|\pi|_{r'_{i,j}} = |\pi|_{r''_{i,j}}$. Without loss of generality we can assume that a rule $r''_{i,j}$ is the next to a rule $r'_{i,j}$ in π (as to the nonterminal $B_{i,j}$ only the rule $r''_{i,j}$ is applicable and we can change the order in which the derivation π is used). Then we can replace any derivation steps of the form $x_1 A_{i,j} x_2 \Rightarrow_{r'_{i,j}} x_1 B_{i,j} x_2 \Rightarrow_{r''_{i,j}} x_1 w_{i,j} x_2$ by $x_1 A_{i,j} x_2 \Rightarrow_{r_{i,j}} x_1 w_{i,j} x_2$.

Accordingly, the occurrence sequence $t'_{i,j} t''_{i,j}$, $\mu \xrightarrow{t'_{i,j}} \mu' \xrightarrow{t''_{i,j}} \mu''$, is replaced by $t_{i,j}$, $\mu \xrightarrow{t_{i,j}} \mu''$, where $t_{i,j} = \gamma'^{-1}(r_{i,j})$, $t'_{i,j} = \gamma'^{-1}(r'_{i,j})$ and $t''_{i,j} = \gamma'^{-1}(r''_{i,j})$, $1 \le i \le 2$, $1 \le j \le k_i$. Clearly, $L(G') \subseteq L(G)$.

Let us consider the general case $k \ge 1$. Let $G = (V, \Sigma, S, R, N_k)$ be a k-Petri net controlled grammar where $N_k = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ is a k-Petri net with $Q = \{q_1, q_2, \ldots, q_k\}$. We can repeat the arguments of the proof for $k = 1$ considering q_k instead of q and adding the new counter place q_{k+1}.

For $k \ge 1$, let the language L_k be defined by

$$
L_k = \{\prod_{i=1}^{k} a_i^{n_i} b_i^{n_i} c_i^{n_i} \mid n_i \ge 1, 1 \le i \le k\}.
$$

Then we can show analogously to Example 3 and Lemma 1 that, for $k \geq 1$,

$$L_{k+1} \in \mathbf{PN}_{k+1} \text{ and } L_{k+1} \notin \mathbf{PN}_k.$$

Thus the inclusions are strict. □

5 Closure Properties

We define the following binary form for k-PN controlled grammars, which will be used in some of the next proofs.

Definition 5. *A k-Petri net controlled grammar $G = (V, \Sigma, S, R, N_k)$ is said to be in a binary form if for each rule $A \to \alpha \in R$, the length of α is not greater than 2, i.e., $|\alpha| \leq 2$.*

Lemma 6 (Binary Form). *For each k-Petri net controlled grammar there exists an equivalent k-Petri net controlled grammar in the binary form.*

Proof. Let $G = (V, \Sigma, S, R, N_k)$ be a k-Petri net controlled grammar with $N_k = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$.

We denote by $R^{>2}$ the set of all rules of the form $A \to \alpha \in R$ where $|\alpha| > 2$.

For each rule $r = A \to x_1 x_2 \cdots x_n \in R^{>2}$, $x_1, x_2, \ldots, x_n \in V \cup \Sigma$ we set

$$V_r = \{B_1, B_2, \ldots, B_{n-2}\}$$

and

$$R_r = \{A \to x_1 B_1, B_1 \to x_2 B_2, \ldots, B_{n-2} \to x_{n-1} x_n\}$$

where B_i, $1 \leq i \leq n-2$, are new nonterminal symbols, $V_r \cap V_{r'} = \emptyset$ for all $r, r' \in R$, $r \neq r'$, and $V_r \cap V = \emptyset$ for all $r \in R$. Let

$$V' = V \cup \bigcup_{r \in R^{>2}} V_r \text{ and } R' = (R \cup \bigcup_{r \in R^{>2}} R_r) - R^{>2}.$$

We define the context-free grammar $G' = (V', \Sigma, S, R')$ and construct a k-Petri net $N'_k = (P', T', F', \varphi', \zeta', \gamma', \mu'_0, \tau')$ with respect to G' such that

(1) for $A \to \alpha \in R$, $|\alpha| \leq 2$,

$$\gamma^{-1}(A \to \alpha) \in {}^\bullet q \cup q^\bullet \text{ iff } \gamma'^{-1}(A \to \alpha) \in {}^\bullet q' \cup q'^\bullet,$$

(2) for $A \to \alpha \in R$, $|\alpha| > 2$,

$$\gamma^{-1}(A \to \alpha) \in {}^\bullet q \text{ iff } \gamma'^{-1}(B_{n-2} \to x_{n-1} x_n) \in {}^\bullet q', \tag{14}$$

$$\gamma^{-1}(A \to \alpha) \in q^\bullet \text{ iff } \gamma'^{-1}(A \to x_1 B_1) \in q'^\bullet \tag{15}$$

where $\alpha = x_1 x_2 \cdots x_n$, $x_i \in V \cup \Sigma$, $1 \leq i \leq n$.

Let $D : S \stackrel{r_1 r_2 \cdots r_k}{\Longrightarrow} w, w \in \Sigma^*$ be a derivation in the grammar G. Then $t_1 t_2 \cdots t_k = \gamma^{-1}(r_1 r_2 \cdots r_k)$ is a successful occurrence sequence of transitions in N_k. We construct a derivation D' in the grammar G' from D as follows.

If for some $1 \leq m \leq k$, $r_m : A \to x_1 x_2 \cdots x_n \in R^{>2}$ then we replace the derivation step

$$y_1 A y_2 \underset{r_m}{\Longrightarrow} y_1 x_1 x_2 \cdots x_n y_2$$

by the derivation steps

$$y_1 A y_2 \underset{r'_1}{\Longrightarrow} y_1 x_1 B_1 y_2 \underset{r'_2}{\Longrightarrow} y_1 x_1 x_2 B_2 y_2 \underset{r'_3}{\Longrightarrow} \cdots \underset{r'_{n-2}}{\Longrightarrow} y_1 x_1 x_2 \cdots x_n y_2$$

where $r'_i \in R_{r_m}$, $1 \leq i \leq n-2$. Correspondingly, $\mu_m \xrightarrow{t_m} \mu_{m+1}$ is replaced by

$$\mu_m \xrightarrow{t'_1 t'_2 \cdots t'_{n-2}} \mu_{m+1}$$

where $t'_i = \gamma'^{-1}(r'_i)$, $1 \leq i \leq n-2$. By (14)–(15), the number of tokens produced and consumed by the transitions $t'_1, t'_2, \ldots, t'_{n-2}$ and the transition t_m are the same. Then D' is a derivation in G', which generates the same word as D does, i.e., $L(G) \subseteq L(G')$.

Inverse inclusion can also be shown using the similar arguments. □

Lemma 7 (Union). *The family of languages* $\mathbf{PN}_k^{[\lambda]}$, $k \geq 1$ *is closed under union.*

Proof. Let $G_1 = (V_1, \Sigma_1, S_1, R_1, N_{k,1})$ and $G_2 = (V_2, \Sigma_2, S_2, R_2, N_{k,2})$ be two k-PN controlled grammars where $N_{k,i} = (P_i \cup Q_i, T_i, F_i \cup E_i, \varphi_i, \zeta_i, \gamma_i, \mu_i, \tau_i)$, $i = 1, 2$ (with the notions of Definition 2). We assume (without loss of generality) that $V_1 \cap V_2 = \emptyset$. We construct the k-PN controlled grammar

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}, N_k)$$

where $N_k = (P, T, F, \varphi, \zeta, \gamma, \mu_0, \tau)$ is defined by

- the set of places: $P = P_1 \cup P_2 \cup Q_1 \cup \{q\}$ where q is a new place;
- the set of transitions: $T = T_1 \cup T_2 \cup \{t_{01}, t_{02}\}$ where t_{01} and t_{02} are new transitions;
- the set of arcs:

$$\begin{aligned} F = F_1 \cup F_2 \cup E_1 \cup \{&(q, t_{0i}), (t_{0i}, p_{0i}) \mid i = 1, 2\} \\ &\cup \{(t, q_{1i}) \mid (t, q_{2i}) \in E_2, 1 \leq i \leq k\} \\ &\cup \{(q_{1i}, t) \mid (q_{2i}, t) \in E_2, 1 \leq i \leq k\} \end{aligned}$$

where p_{0i} are the places labeled by S_i, i.e., $\zeta_i(p_{0i}) = S_i$, $i = 1, 2$;

- the weight function:

$$\varphi(x,y) = \begin{cases} \varphi_i(x,y) & \text{if } (x,y) \in F_i, i = 1,2, \\ 1 & \text{otherwise;} \end{cases}$$

- the labeling function ζ is defined by

$$\zeta(p) = \begin{cases} \zeta_1(p) & \text{if } p \in P_1 \cup Q_1, \\ \zeta_2(p) & \text{if } p \in P_2 \\ S & \text{if } p = q; \end{cases}$$

- the labeling function γ is defined by

$$\gamma(t) = \begin{cases} \gamma_i(t) & \text{if } t \in T_i, i = 1,2, \\ S \to S_i & \text{if } t = t_{0i}, i = 1,2; \end{cases}$$

- the initial marking:

$$\mu_0(p) = \begin{cases} 1 & \text{if } p = q, \\ 0 & \text{otherwise;} \end{cases}$$

- the final marking: $\tau(p) = 0$ for all $p \in P$.

By the construction of N_k any occurrence of its transitions can start by firing of t_{01} or t_{02} then transitions of T_1 or transitions of T_2 can occur, correspondingly we start a derivation with the rule $S \to S_1$ or $S \to S_2$ then we can use rules of R_1 or R_2.

A string w is in $L(G)$ if and only if there is a derivation $S \Rightarrow S_i \Rightarrow^* w \in L(G_i)$, $i = 1,2$. On the other hand, we can initialize any derivation $S_i \Rightarrow^* w \in L(G_i)$ with the rule $S \to S_i$, $i = 1,2$, i.e., $w \in L(G)$. □

Lemma 8 (Concatenation). *The family of languages* $\mathbf{PN}_k$, $k \geq 1$ *is not closed under concatenation.*

Proof. Let L_k and L'_k be two languages, with the same structure but disjoint alphabets, given at the end of the proof of Theorem 1. Then $L_k, L'_k \in \mathbf{PN}_k$ and $L_k \cdot L'_k \notin \mathbf{PN}_k$. □

The next lemma shows that the concatenation of two languages generated by k- and m-PN controlled grammars, $k, m \geq 1$, can be generated by a $(k+m)$-PN controlled grammar.

Lemma 9. *For* $L_1 \in \mathbf{PN}_k^{[\lambda]}$, $k \geq 1$ *and* $L_2 \in \mathbf{PN}_m^{[\lambda]}$, $m \geq 1$,

$$L_1 \cdot L_2 \in \mathbf{PN}_{k+m}^{[\lambda]}.$$

Proof. Let $G_1 = (V_1, \Sigma, S_1, R_1, N_k)$ where $N_k = (P_1, T_1, F_1, \varphi_1, \zeta_1, \gamma_1, \mu_1, \tau_1)$ and $G_2 = (V_2, \Sigma, S_2, R_2, N_m)$ where $N_m = (P_2, T_2, F_2, \varphi_2, \zeta_2, \gamma_2, \mu_2, \tau_2)$ be, respectively, k-Petri net and m-Petri net controlled grammars such that $L(G_1) = L_1$ and $L(G_2) = L_2$. Without loss of generality we assume that $V_1 \cap V_2 = \emptyset$. We set $V = V_1 \cup V_2 \cup \{S\}$ where S is a new nonterminal and

$$R = R_1 \cup R_2 \cup \{S \to S_1 S_2\}.$$

We define a $(k+m)$-PN controlled grammar $G = (V, \Sigma, S, R, N_{k+m})$ with $N_{k+m} = (P, T, F, \varphi, \zeta, \gamma, \mu_0, \tau)$ where

- $P = P_1 \cup P_2 \cup \{p_0\}$ where p_0 is a new place;
- $T = T_1 \cup T_2 \cup \{t_0\}$ where t_0 is a new transition;
- $F = F_1 \cup F_2 \cup \{(p_0, t_0), (t_0, p_1), (t_0, p_2)\}$ where $\zeta_i(p_i) = S_i$, $i = 1, 2$;
- the weight function φ is defined by

$$\varphi(x, y) = \begin{cases} \varphi_i(x, y) & \text{if } (x, y) \in F_i, i = 1, 2, \\ 1 & \text{otherwise;} \end{cases}$$

- the labeling function ζ is defined by

$$\zeta(p) = \begin{cases} \zeta_i(p) & \text{if } p \in P_i, i = 1, 2, \\ S & \text{if } p = p_0; \end{cases}$$

- the labeling function γ is defined by

$$\gamma(t) = \begin{cases} \gamma_i(t) & \text{if } t \in T_i, i = 1, 2, \\ S \to S_1 S_2 & \text{if } t = t_0; \end{cases}$$

- the initial marking:

$$\mu_0(p) = \begin{cases} 1 & \text{if } p = p_0, \\ 0 & \text{otherwise;} \end{cases}$$

- the final marking: $\tau(p) = 0$ for all $p \in P$.

It is not difficult to see that $L(G) = L(G_1)L(G_2)$. □

Lemma 10 (Substitution). *The family of languages* $\mathbf{PN}_k$, $k \geq 1$ *is closed under substitution by context-free languages.*

Proof. Let $G = (V, \Sigma, S, R, N_k)$ be a k-PN controlled grammar with k-Petri net $N_k = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$. We consider a substitution $s : \Sigma^* \to 2^{\Delta^*}$ with $s(a) \in \mathbf{CF}$ for each $a \in \Sigma$. Let $G_a = (V_a, \Sigma_a, S_a, R_a)$ be a context-free grammar for $s(a)$, $a \in \Sigma$. We can assume that $V \cap V_a = \emptyset$ for any $a \in \Sigma$ and $V_a \cap V_b = \emptyset$ for any $a, b \in \Sigma$, $a \neq b$.

Let $N_a = (P_a, T_a, F_a, \phi_a, \beta_a, \gamma_a, \iota_a)$ be a cf Petri net with respect to the grammar $G_a, a \in \Sigma$. We define the k-PN controlled grammar

$$G' = (V \cup \Sigma \cup \bigcup_{a \in \Sigma} V_a, \Delta, S, R' \cup \bigcup_{a \in \Sigma} R_a, N'_k)$$

where R' is the set of rules obtained by replacing each occurrence of $a \in \Sigma$ by S_a in R and N'_k is defined by

$$N'_k = (P \cup Q \cup P_\Sigma \cup \bigcup_{a \in \Sigma} P_a, T \cup \bigcup_{a \in \Sigma} T_a, F \cup F_\Sigma \cup \bigcup_{a \in \Sigma} F_a, \varphi', \zeta', \gamma', \mu'_0, \tau')$$

where

- $P_\Sigma = \{p_a \mid a \in \Sigma\}$ is the set of new places;
- $F_\Sigma = \{(t, p_a) \mid \gamma(t) = A \to \alpha, |\alpha|_a > 0, a \in \Sigma\}$ is the set of new arcs;
- the weight function φ' is defined by

$$\varphi'(x,y) = \begin{cases} \varphi(x,y) & \text{if } (x,y) \in F, \\ \phi_a(x,y) & \text{if } (x,y) \in F_a, a \in \Sigma, \\ |\alpha|_a, & \text{if } x = t, y = p_a, (t, p_a) \in F_\Sigma, a \in \Sigma; \end{cases}$$

- the labeling function ζ' is defined by

$$\zeta'(p) = \begin{cases} \zeta(p) & \text{if } p \in (P \cup Q), \\ \beta_a(p) & \text{if } p \in P_a, a \in \Sigma, \\ S_a & \text{if } p = p_a \in P_\Sigma, a \in \Sigma; \end{cases}$$

- the labeling function γ' is defined by

$$\gamma'(t) = \begin{cases} \gamma(t) & \text{if } t \in T, \\ \gamma_a(t) & \text{if } t \in T_a, a \in \Sigma; \end{cases}$$

- the initial marking:

$$\mu'_0(p) = \begin{cases} 1 & \text{if } p = \zeta'^{-1}(S), \\ 0 & \text{otherwise;} \end{cases}$$

- the final marking: $\tau'(p) = 0$ for all $p \in P'$;

Obviously, $L(G') \in \mathbf{PN}_k$. □

Lemma 11 (Mirror Image). *The family of languages* $\mathbf{PN}_k$, $k \geq 1$ *is closed under mirror image.*

Proof. Let $G = (V, \Sigma, S, R, N_k)$ be a k-PN controlled grammar. Let

$$R^- = \{A \to x_n \cdots x_2 x_1 \mid A \to x_1 x_2 \cdots x_n \in R\}.$$

The context-free grammar (V, Σ, S, R) and its reversal (V, Σ, S, R^-) have the same corresponding cf Petri net $N = (P, T, F, \phi, \beta, \gamma, \iota)$ as N does not preserve the order of the positions of the output places for each transition. Thus we can also use the k-Petri net N_k as a control mechanism for the grammar (V, Σ, S, R^-), i.e. we define $G^- = (V, \Sigma, S, R^-, N_k)$. Clearly, $L(G^-) \in \mathbf{PN}_k$. □

Lemma 12 (Intersection with Regular Languages). *The family of languages* $\mathbf{PN}_k$, $k \geq 1$ *is closed under intersection with regular languages.*

Proof. We use the arguments and notions of the proof of Lemma 1.3.5 in [2]. Let $G = (V, \Sigma, S, R, N_k)$ be a k-Petri net controlled grammar with a k-Petri net $N_k = (P \cup Q, T, F \cup E, \varphi, \zeta, \gamma, \mu_0, \tau)$ (with the notions of Definition 2). Without loss of generality we can assume that G is in a binary form.

Let $\mathcal{A} = (K, \Sigma, s_0, \delta, H)$ be a deterministic finite automaton. We set

$$V' = \{[s, x, s'] \mid s, s' \in K,\ x \in V \cup \Sigma\}.$$

For each rule $r \in R$ we construct the set $R(r)$ in the following way

1. If $r = A \to x_1 x_2,\ x_1, x_2 \in V \cup \Sigma$ then

$$R(r) = \{[s, A, s'] \to [s, x_1, s'][s', x_2, s''] \mid s, s', s'' \in K\}.$$

2. If $r = A \to x,\ x \in V \cup \Sigma$ then

$$R(r) = \{[s, A, s'] \to [s, x, s'] \mid s, s' \in K\}.$$

Further we define the set of rules

$$R_\Sigma = \{[s, a, s'] \to a \mid s' = \delta(s, a), s, s' \in K, a \in \Sigma\}.$$

Let

$$R' = \bigcup_{r \in R} R(r) \cup R_\Sigma.$$

We define the context-free grammar $G_s = (V', \Sigma, [s_0, S, s], R')$ for each $s \in H$. Let $N_s = (P_s, T_s, F_s, \phi_s, \beta_s, \gamma_s, \iota_s)$ be a cf Petri net with respect to the grammar G_s where

$$P_s = \{[s, p, s'] \mid s, s' \in K, p \in P\},$$
$$T_s = \{[s, t, s'] \mid s, s' \in K, p \in P\},$$
$$F_s = \{([s_1, x, s_2], [s_1', y, s_2']) \mid s_1, s_2, s_1', s_2' \in K, (x, y) \in F\}.$$

The weight function ϕ_s is defined by $\phi([s_1, x, s_2], [s_1', y, s_2']) = \phi(x, y)$ where $s_1, s_2, s_1', s_2' \in K, (x, y) \in F$.

The functions $\beta_s : P_s \to V'$ and $\gamma_s : T_s \to R'$ are bijections, and

$$\iota_s(\beta_s^{-1}([s_0, S, s])) = 1 \text{ and } \iota_s(p) = 0 \text{ for all } P_s - \{\beta_s^{-1}([s_0, S, s])\}.$$

We set

$$F_Q^- = \{((s, t, s'), q) \mid s, s' \in K, q \in Q \wedge t \in {}^\bullet q\}$$

and

$$F_Q^+ = \{(q, (s, t, s')) \mid s, s' \in K, q \in Q \wedge t \in q^\bullet\}.$$

We construct the k-Petri net

$$N_{k,s} = (P_s \cup Q, T_s, F_s \cup F_Q^- \cup F_Q^+, \varphi_s, \zeta_s, \gamma_s, \mu_s, \tau_s)$$

from N_s where

- the weight function φ_s is defined by

$$\varphi_s([s_1, x, s_2], [s_1', y, s_2']) = \varphi(x, y), s_1, s_1', s_2, s_2' \in K \text{ and } (x, y) \in F \cup E,$$

- the labeling function ζ_s is defined by

$$\zeta_s([s_1, p, s_2]) = \begin{cases} \beta_s([s_1, p, s_2]) & \text{if } [s_1, p, s_2] \in P_s, \\ \lambda & \text{if } [s_1, p, s_2] \in Q, \end{cases}$$

- the initial marking μ_s is defined by $\mu_s(\beta_s^{-1}([s_0, S, s])) = 1$ and $\mu_s(p) = 0$ for all $(P_s \cup Q) - \{\beta_s^{-1}([s_0, S, s])\}$,

- the final marking τ_s is defined by $\tau_s(p) = 0$ for all $p \in P_s \cup Q$,

and define the k-PN controlled grammar $G_s' = (V', \Sigma, (s_0, S, s), R', N_{k,s})$. Then one can see that $L(G) \cap L(A) = \bigcup_{s \in H} L(G_s')$. □

The results of the previous lemmas are summarized in the following theorem:

Theorem 2. *The family of languages* $\mathbf{PN}_k$, $k \geq 1$, *is closed under union, substitution, mirror image, intersection with regular languages and it is not closed under concatenation.*

References

[1] Crespi-Reghizzi, S. and Mandrioli, D. Petri nets and commutative grammars. Internal Report 74-5, Laboraterio di Calcolatori, Instituto di Elettrotecnica ed Elettromca del Politecnico di Milano, Italy, 1974.

[2] Dassow, J. and Păun, Gh. *Regulated rewriting in formal language theory.* Springer-Verlag, Berlin, 1989.

[3] Dassow, J. and Turaev, S. k-Petri net controlled grammars. In Martín-Vide, C., Otto, F., and Fernau, H., editors, *Language and Automata Theory and Applications. Second International Conference, LATA 2008. Revised Papers*, volume 5196 of *LNCS*, pages 209–220. Springer, 2008.

[4] Esparza, J. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundam. Inf.*, 31(1):13–25, 1997.

[5] Hack, M. *Decidablity questions for Petri nets.* PhD thesis, Dept. of Electrical Engineering, MIT, 1975.

[6] Hopcroft, J.E. and Ullman, J.D. *Introduction to automata theory, languages, and computation.* Addison-Wesley Longman Publishing Co., Inc., 1990.

[7] Marek, V. and Češka, M. Petri nets and random-context grammars. In *Proc. of the 35th Spring Conference: Modelling and Simulation of Systems*, pages 145–152, MARQ Ostrava, Hardec nad Moravicí, 2001.

[8] Martín-Vide, C., Mitrana, V., and Păun, Gh., editors. *Formal languages and applications.* Springer-Verlag, Berlin, 2004.

[9] Murata, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[10] Petri, C.A. *Kommunication mit Automaten.* PhD thesis, University of Bonn, 1962.

[11] Reisig, W. and Rozenberg, G., editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, Berlin, 1998. Springer.

[12] Starke, P.H. *Petri-Netze.* Deutscher Verlag der Wissenschaften, 1980.

[13] Turaev, S. Petri net controlled grammars. In *Third Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2007*, pages 233–240, Znojmo, Czechia, 2007.

Received 26th January 2009

Acta Cybernetica **19** (2010) 635–653.

Modeling a Domain in a Tutorial-*like* System Using Learning Automata*

B. John Oommen† and M. Khaled Hashem‡

Abstract

The aim of this paper is to present a novel approach to model a knowledge domain for teaching material in a Tutorial-*like* system. In this approach, the Tutorial-*like* system is capable of presenting teaching material within a Socratic model of teaching. The corresponding questions are of a multiple choice type, in which the complexity of the material increases in difficulty. This enables the Tutorial-*like* system to present the teaching material in different chapters, where each chapter represents a level of difficulty that is harder than the previous one. We attempt to achieve the entire learning process using the Learning Automata (LA) paradigm. In order for the Domain model to possess an increased difficulty for the teaching Environment, we propose to correspondingly reduce the *range* of the penalty probabilities of all actions by incorporating a scaling factor μ. We show that such a scaling renders it more difficult for the Student to infer the correct action within the LA paradigm.

To the best of our knowledge, the concept of modeling teaching material with increasing difficulty using a LA paradigm is unique. The main results we have obtained are that increasing the difficulty of the teaching material can affect the learning of Normal and Below-Normal Students by resulting in an increased learning time, but it seems to have no effect on the learning behavior of Fast Students. The proposed representation has been tested for different benchmark Environments, and the results show that the difficulty of the Environments can be increased by decreasing the range of the penalty probabilities. For example, for some Environments, decreasing the range of the penalty probabilities by 50% results in increasing the difficulty of learning for Normal Students by more than 60%.

Keywords: domain modeling, tutorial-like systems, learning automata, mod-

*The first author was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada. A preliminary version of this paper was presented at the Proceedings of ICMLC 2007, the 2007 International Conference of Machine Learning and Cybernetics, Hong Kong, China, August 2007.

†*Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway. oommen@scs.carleton.ca.

‡This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6. k_hashem@yahoo.com.

eling of adaptive systems.

1 Introduction

Representing the domain knowledge in Tutorials systems, in an effective way, is a challenging task. This includes how the knowledge is represented, and how it should be structured so as to reflect the nature of increasing the complexity/difficulty of the material to be taught in the Tutorial system [4]. The Domain model is, typically, the model that permits such a customization, and is usually application dependent.

From a systems perspective, the Domain model is the control center that encompasses the entire domain knowledge. The Teacher utilizes the domain knowledge as represented in the Domain model. Further, he[1] incorporates it into his teaching model to present the material to the Students in a manner that is customizable to each Student.

Analogous to traditional Tutorial systems, our Tutorial-*like* system utilizes the Domain model to represent the domain knowledge in a manner that enables the Teacher to conduct the learning to the Students in an effective way. The aim of this paper is to present how the domain knowledge can be modeled and implemented in such Tutorial-*like* systems. The domain knowledge is presented using a Socratic model and *via* multiple choice questions within the Learning Automata (LA) paradigm. For each question, every choice has an associated probability that this choice is correct, and the answer to any specific question is the choice with the highest reward probability.

Our model utilizes concepts from the field of LA, where the Domain model incorporates a novel mechanism to present the teaching material. The salient features of this mechanism can be summarized as follows:

- The knowledge is presented via multiple choice questions, which also serve to *test* the learning mechanism.
- The collection/set of questions also constitutes a chapter in the knowledge to be imparted.
- Subsequent chapters are more difficult than preceding ones.
- The answers to the question for subsequent chapters are not predictable by virtue of prior knowledge.

All of these details will be clarified presently.

In order for the Domain model to render a question to be more difficult, we propose that it reduces the so-called penalty probabilities for the choices pertinent to that question. This makes the Environment's response to the choices of that question less predictable. The experimental results of our model, as will be presented

[1] For the case of communication, we request the permission to refer to the entities involved (i.e. the Teacher, Student, etc.) in the masculine.

later in the paper, demonstrate that this approach is feasible in representing the knowledge in a Tutorial-*like* system. The approach has been tested in benchmark Environments and the results are both intuitively appealing and rather fascinating considering that the Students and Teacher are not *real-life* entities, but rather, "models". Increasing the difficulty of the teaching environment proved to make the learning more difficult for Normal and Below-Normal Students, while Fast-learning Students were apparently not adversely effected by the increased difficulty. For example, for Below-Normal learners, when the range of the reward probabilities was decreased by 50% in the benchmark Environments, the difficulty within the teaching Environments increased by more than 60%. We believe that our Domain model representation is a novel approach within the field of LA modeling, which permits the LA Environments in this field to consistently increase their difficulties so as to mimic the teaching of material with increasing complexities.

The different components of Tutorial-*like* systems are also modeled, namely, the Student model, the Teacher model, and the Student-Classroom interaction model. These models have been studied in detail elsewhere [15, 14], but briefly mentioned here to permit readability.

1.1 Tutorial-*like* Systems

Our entire research will be within the context of Tutorial-*like* systems [14]. In these systems, there need not be *real-life* Students, but rather each Student could be replaced by a Student Simulator that mimics a *real-life* Student. Alternatively, it could also be a software entity that attempts to learn. The Teacher, in these systems, attempts to present the teaching material to a *School* of Student Simulators. The Students (synonymously referred to also as Student Simulators) are also permitted to share information between each other to gain knowledge. Therefore, such a teaching environment allows the Students to gain knowledge not only from the Teacher but also from other fellow Students.

In the Tutorial-*like* systems which we study, the Teacher has a *stochastic* nature, where he has an imprecise knowledge of the material to be imparted. The Teacher also doesn't have a prior knowledge about how to teach the subject material. He "learns" that himself while using the system, and thus, hopefully, improves his skills as a teacher. Observe that, conceptually, the Teacher, in some sense, is also a "student".

On the other hand, the Student Simulators need to learn from the Stochastic Teacher, as well as from each other. Each Student needs to decide when to request assistance from a fellow Student and how to "judge" the quality of information he receives from them. Thus, we require each Student to possess a mechanism whereby it can detect a scenario of procuring inaccurate information from other Students.

In our model of teaching/learning, the teaching material of the Tutorial-*like* system follows a Socratic model, where the domain knowledge is represented in the form of questions, either to be of a *Multiple Choice* sort or, in the most extreme case, of a *Boolean* sort. These questions, in our present paradigm, carry some degree of uncertainty, where each question has a probability that indicates the accuracy for

the answer of that question.

1.2 Stochastic Learning Automaton

Learning Automaton[2] (LA) have been used in systems that have incomplete knowledge about the Environment in which they operate [1, 17, 20, 21, 23, 30, 37]. The learning mechanism attempts to learn from a *stochastic Teacher* which models the Environment. In his pioneer work, Tsetlin [38] attempted to use LA to model biological learning. In general, a random action is selected based on a probability vector, and these action probabilities are updated based on the observation of the Environment's response, after which the procedure is repeated.

The term "Learning Automata" was first publicized in the survey paper by Narendra and Thathachar. The goal of LA is to "determine the optimal action out of a set of allowable actions" [1]. The distinguishing characteristic of automata-based learning is that the search for the optimizing parameter vector is conducted in the space of probability distributions defined over the parameter space, rather than in the parameter space itself [36].

In the first LA designs, the transition and the output functions were time invariant, and for this reason these LA were considered "fixed structure" automata. Tsetlin, Krylov, and Krinsky [38] presented notable examples of this type of automata. Later, Vorontsova and Varshavskii introduced a class of stochastic automata known in the literature as Variable Structure Stochastic Automata (VSSA). In the definition of a VSSA, the LA is completely defined by a set of actions (one of which is the output of the automaton), a set of inputs (which is usually the response of the Environment) and a learning algorithm, T. The learning algorithm [21] operates on a vector (called *the Action Probability vector*)

$P(t) = [p_1(t), \ldots, p_r(t)]^T$,

where $p_i(t)$ ($i = 1, \ldots, r$) is the probability that the automaton will select the action α_i at time 't',

$p_i(t) = Pr[\alpha(t) = \alpha_i]$, $i = 1, \ldots, r$, and it satisfies

$\sum_{i=1}^{r} p_i(t) = 1 \; \forall \; t$.

Note that the algorithm $T : [0,1]^r \times A \times B \rightarrow [0,1]^r$ is an updating scheme where $A = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$, $2 \leq r < \infty$, is the set of output actions of the automaton, and B is the set of responses from the Environment. Thus, the updating is such that

$P(t+1) = T(P(t), \alpha(t), \beta(t))$,

where $P(t)$ is the action probability vector, $\alpha(t)$ is the action chosen at time t, and $\beta(t)$ is the response it has obtained.

If the mapping T is chosen in such a manner that the Markov process has absorbing states, the algorithm is referred to as an absorbing algorithm. Many families of VSSA that posses absorbing barriers have been reported [21]. Ergodic VSSA have also been investigated [21, 25]. These VSSA converge in distribution and thus, the asymptotic distribution of the action probability vector has a value

[2]In the interest of completeness, we have included a fairly good review of the field of LA here. This can be deleted or abridged as per the desire of the Referees.

that is independent of the corresponding initial vector. Thus, while ergodic VSSA are suitable for non-stationary environments, automata with absorbing barriers are preferred in stationary environments.

In practice, the relatively slow rate of convergence of these algorithms constituted a limiting factor in their applicability. In order to increase their speed of convergence, the concept of discretizing the probability space was introduced [25, 35]. This concept is implemented by restricting the probability of choosing an action to a finite number of values in the interval [0,1]. If the values allowed are equally spaced in this interval, the discretization is said to be linear, otherwise, the discretization is called non-linear. Following the discretization concept, many of the continuous VSSA have been discretized; indeed, discrete versions of almost all continuous automata have been presented in the literature [25].

Pursuit and Estimator-based LA were introduced to be faster schemes, characterized by the fact that they pursue what can be reckoned to be the *current* optimal action or the set of current optimal schemes [25]. The updating algorithm improves its convergence results by using the history to maintain an estimate of the probability of each action being rewarded, in what is called the *reward-estimate* vector. While, in non-estimator algorithms, the action probability vector is updated solely on the basis of the Environment's response, in a Pursuit or Estimator-based LA, the update is based on *both* the Environment's response and the *reward-estimate* vector. Families of Pursuit and Estimator-based LA have been shown to be faster than VSSA [36]. Indeed, even faster discretized versions of these schemes have been reported [1, 25].

With regard to applications, the entire field of LA and stochastic learning, has had a myriad of applications [17, 20, 21, 30, 37], which (apart from the many applications listed in these books) include solutions for problems in network and communications [19, 22, 27, 29], network call admission, traffic control, quality of service routing, [2, 3, 40], distributed scheduling [34], training hidden Markov models [16], neural network adaptation [18], intelligent vehicle control [39], and even fairly theoretical problems such as graph partitioning [26]. Besides these fairly generic applications, with a little insight, LA can be used to assist in solving (by, indeed, learning the associated parameters) the stochastic resonance problem [10], the stochastic sampling problem in computer graphics [11], the problem of determining roads in aerial images by using geometric-stochastic models [6], the stochastic and dynamic vehicle routing problem [7], and various location problems [9]. Similar learning solutions can also be used to analyze the stochastic properties of the random waypoint mobility model in wireless communication networks [8], to achieve spatial point pattern analysis codes for GISs [31], to digitally simulate wind field velocities [28], to interrogate the experimental measurements of global dynamics in magneto-mechanical oscillators [12], and to analyze spatial point patterns [5]. LA-based schemes have already been utilized to learn the best parameters for neural networks [18], optimizing QoS routing [41], and bus arbitration [22] – to mention a few other applications.

1.3 Contributions of this paper

This paper presents a novel approach to model the domain knowledge in a Tutorial-*like* system. The representation of the knowledge can be crucial in building effective Tutorial systems. Thus, the salient contributions of this paper are as follows:

- The modeling of the domain knowledge using an LA paradigm using a Socratic model.
- Questions, in this model, are represented to be of a multiple-choice type, with stochastic solutions.
- The Student needs to learn the domain knowledge by responding to the questions.
- Enabling the Domain model to increase the complexity/difficulty of the domain knowledge.
- The knowledge is presented in chapters, where subsequent chapters are more difficult than preceding ones.

2 Intelligent Tutorial and Tutorial-*like* Systems

Since our research involves Tutorial-*like* systems, which are intended to mimic Tutorial systems, a brief overview of these follows.

Intelligent Tutorial Systems (ITSs) are special educational software packages that involve Artificial Intelligence (AI) techniques and methods to represent the knowledge, as well as to conduct the learning interaction [24]. ITSs are characterized by their responsiveness to the learner's need. They adapt according to the knowledge/skill of the users. They also incorporate experts' domain specific knowledge.

An ITS mainly consists of a number of modules, typically three [13], and sometimes four when a communication module (interface) is added [42]. The former three modules are the domain model (knowledge domain), the student model, and the pedagogical model, (which represent the tutor model itself). Self [33] defined these components as the tripartite architecture for an ITS – the *what* (domain model), the *who* (student model), and the *how* (tutoring model). Figure 1 depicts a common ITS architecture.

2.1 Tutorial-*like* Systems

Tutorial-*like* systems share some similarities with the well-developed field of Tutorial systems. Thus, for example, they model the Teacher, the Student, and the Domain knowledge. However, they are different from "traditional" Tutorial systems in the characteristics of their models, etc. as will be highlighted below.

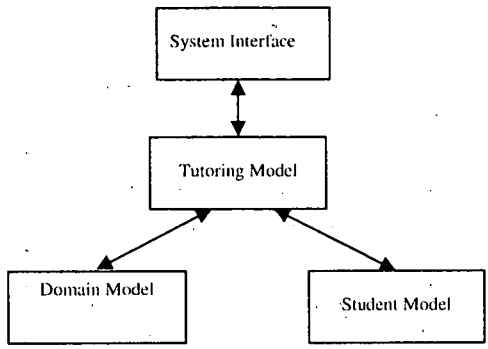

Figure 1: A Common ITS Architecture.

1. **Different Type of Teacher.** In Tutorial systems, as they are developed today, the Teacher is assumed to have perfect information about the material to be taught. Also, built into the model of the Teacher is the knowledge of how the domain material is to be taught, and a plan of how it will communicate and interact with the Student(s). This teaching strategy may progress and improve over time. The Teacher in our Tutorial-*like* system possesses different features. First of all, one fundamental difference is that the Teacher is uncertain of the teaching material – he is stochastic. Secondly, the Teacher does not *initially* possess any knowledge about "How to teach" the domain subject. Rather, the Teacher himself is involved in a "learning" process and he "learns" what teaching material has to be presented to the particular Student. To achieve this, as mentioned, we assume that the Teacher follows the Socratic model of learning by teaching the material using questions that are presented to the Students. He then uses the feedback from the Students and their corresponding LA to suggest new teaching material.

   Although removing the "How to teach" knowledge from the Teacher would take away the "bread and butter" premise of the teaching process in a Tutorial system, in a Tutorial-*like* system, removing this knowledge allows the system to be modeled without excessive complications, and renders the modeling of knowledge less burdensome. The success of our proposed methodology would be beneficial to systems in which any domain knowledge pertinent to tutoring teaching material could be merely plugged into the system without the need to worry about "how to teach" the material.

2. **No Real Students.** A Tutorial system is intended for the use of *real-life* students. Its measure of accomplishment is based on the performance of these students after using the system, and it is often quantified by comparing their progress with other students in a control group, who would use a *real-life* Tutor. In our Tutorial-*like* system, there are no *real-life* students who use the system. The system could be used by either:

a) Students Simulators, that mimic the behavior and actions of *real-life* students using the system. The latter would themselves simulate how the Students improve their knowledge and their interaction with the Teacher and with other Students. They can also take proactive actions interacting with the teaching environment by one of the following measures:
   i. Asking a question to the Teacher
   ii. Asking a question to another Student
   iii. Proposing to help another Student

b) An artificial Entity which, in itself, could be another software component that needs to "learn" specific domain knowledge.

3. **Uncertain Course Material.** Unlike the domain knowledge of "traditional" Tutorial systems where the knowledge is, typically, well defined, the domain knowledge teaching material presented in our Tutorial-*like* system contains material that has some degree of uncertainty. The teaching material contains questions, each of which has a probability that indicates the certainty of whether the answer to the question is in the affirmative.

4. **Testing Vs. Evaluation.** Sanders [32] differentiates between the concepts of "teaching evaluation" and "teaching testing". He defines "teaching evaluation" as an "interpretive process", in which the Teacher "values, determines merit or worth of the students performance, and their needs". He also defines "teaching testing" as a "data collection process". In a Tutorial system, an evaluation is required to measure the performance of the Student while using the system and acquiring more knowledge. In our Tutorial-*like* system, the Student(s) acquire knowledge using a Socratic model, where it gains knowledge from answering questions without having any prior knowledge about the subject material. In our model, the testing will be based on the performance of the set of Student Simulators.

5. *School* **of Students.** Traditional Tutorial Systems deal with a Teacher who teaches Students, but they do not permit the Students to interact with each other. A Tutorial-*like* system assumes that the Teacher is dealing with a *School* of Students where each learns from the Teacher on his own, and can also learn from his "colleagues" if he desires, or is communicating with a cooperating colleague. Notice that we will have to now consider how each Student learns, and also how the entire *School* learns.

3 Learning of Students in a LA Teaching Environment

In Tutorial-*like* systems, Students (or Student Simulators) try to learn some domain knowledge from the Teacher and from the interaction between themselves. As mentioned earlier, there are no *real-life* Students who use the Tutorial-*like* systems.

Students are modeled using Student Simulators, that try to mimic the actions and behavior of *real-life* Students. Student Simulators are, in turn, modeled using LA which attempt to learn the domain knowledge from the Teacher, who also may be a modeled entity.

First of all, the Tutorial-*like* system models the Students by observing their behavior while using the system and examining how they learn. The Student modeler tries to infer what *type* of Student it is providing the knowledge to. This enables the Teacher to customize his teaching experience to each Student according to his caliber.

If we are dealing with *real-life* Students, it would have been an easy task to implement these concepts in a real Tutorial system. But since the goal of the exercise is to achieve a teaching-learning experience, in which every facet of the interaction involves a model (or a non *real-life* Student), the design and implementation of the entire Tutorial-*like* system must be couched in a formal established learning paradigm. As mentioned earlier, although there are host of such learning methodologies, we have chosen to work within the LA paradigm, as explained in Section 1.2. Thus, the questions encountered, before this endeavor is successful, involve:

- How can we model the Teacher in terms of an LA Environment?
- How can we model the different types of Students that could be involved in the learning?
- How can we model the Domain, from which the learning material is presented?
- How can we model *chapters* of teaching material with increasing complexity?

We shall address all of these issues now, and report the experimental results obtained from such a modeling exercise.

Modeling The Student: In our model, typically, a Student can be one of these three types (although it is easy to generalize this to a larger spectrum of Students):

- Fast Student. This type of Students can be simulated using a Pursuit scheme, which is, typically, a fast convergence scheme.
- Normal Student. The Student Simulator can mimic this type of Students using a VSSA scheme.
- Slow Student. Such a Student can be implemented using a FSSA, or a VSSA with a lower value of λ.

Modeling the Choices: The Tutorial-*like* system uses the Socratic model of teaching by presenting multiple-choice questions to the Students. The Student selects an option from the set of available actions $\underline{\alpha}$, in which α_i is the action corresponding to selecting choice 'i' in the multiple-choice question.

Modeling the stochastic Teacher: The Teacher who imparts the domain knowledge is modeled as an LA Environment, which possesses a set of penalty

probabilities $\underline{c}$, in which c_i is the penalty probability associated with the fact that the Environment penalizes choice 'i'. The Student is unaware of the values of these penalty probabilities.

Modeling the Rewards/Penalties: When the Student selects an action α_i, the Environment can either reward ($\beta = 0$) or penalize ($\beta = 1$) his actions. This feedback provides the Student the information required to learn from his actions, and from this feedback loop, the cycle is repeated. The Student can incrementally learn until his LA, hopefully, converges to the *best* action, which is the one which has the minimum penalty probability.

The crucial issue that has not been addressed as yet is that of modeling the domain itself to consider "chapters" of increasing complexity. That will be addressed and formalized in the next section.

4 Domainsem/Teaching Material with Increasing Difficulty

The Teaching material in the Tutorial-*like* system is modeled as a Socratic model that contains multiple choice questions. Each choice has an associated probability that represents the probability that this choice is correct. Each question is represented using an LA Environment, where the Environment has a penalty probability associated with that choice.

When the Domain model is required to increase the complexity of a question, we propose that it reduces the penalty[3] probabilities of the choices for that question with a scale factor, μ. This results in the reduction of the range of all the penalty probabilities, which makes it more difficult for the Student to determine the *best* choice for the question, primarily because the reduced penalty probabilities tend to cluster together. This is the primary hypothesis of our model, and this will be demonstrated presently.

Formally, the Domain model for the teaching Environment is as follows:

$\{\underline{\alpha}, \underline{\beta}, \underline{c}, \mu\}$, where:

$\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_R\}$, in which

α_i is the action corresponding to selecting choice 'i' in the multiple choice question.

$\underline{\beta} = \{0, 1\}$, in which

$\beta = 0$ implies a Reward for the present action (i.e, choice) chosen, and

$\beta = 1$ implies a Penalty for the present action (i.e, choice) chosen.

$\underline{c} = \{c_1, c_2, \ldots, c_R\}$, in which

c_i is the penalty probability associated with the fact that the Environment will penalize choice 'i'.

μ ($0 < \mu \leq 1$) is the scaling factor which is used to control the complexity/difficulty of any question. The value of μ=1.0 represents a question with a "normal" difficulty, while the difficulty increases as μ decreases.

[3] It is easy to see that a similar scaling can be achieved by manipulating the *reward* probabilities.

The Domain model increases the complexity of the domain knowledge without changing the order of the choices to the question. If the Student is permitted to remember the choices of previous questions, he is indirectly given prior knowledge about the optimal answer to the particular question at hand. However, for the present we assume that as far as the Student is concerned, the chapter presented by the Environment is not related to a previous one[4].

5 Experimental Results

This section presents the experimental results obtained by using the proposed Domain model to represent the teaching material, and to increase its complexity. Numerous simulations were performed in order to study how the knowledge could be modeled, and how the difficulty of the teaching material led to increase the learning time for the different types of Students.

The simulations were performed for different benchmark Environments, two 4-action Environments and two 10-action Environments. The Environments contained multiple choice questions that represented the teaching material that had to be taught to the Students. In these experiments, an algorithm was considered to have converged, if the probability of choosing an action was greater than or equal to a threshold $T(0 < T \leq 1)$. An automaton was considered to converge correctly if it converged to the best choice (i.e. to the action with the highest probability of being rewarded).

The simulations were performed against three types of Students, who communicated with the Teacher and learned the teaching material as follows:

- Fast learning Students. In order for the Student Simulator to mimic Students of this type, we used a Pursuit PL_{RI} scheme, with λ being in the range 0.0041 to 0.0127. In this scheme, the action probability vector is updated if the LA obtained a reward. As for the Pursuit algorithm, the estimate vector for the reward probabilities was always updated.

- Normal learning Students. In this case, we used a VSSA to simulate Students of this type. In particular, we utilized the L_{RI} scheme with λ being in the range 0.0182 to 0.0192.

- Below-Normal Students ("Slow Learners"). In this case too, the Student Simulators used VSSA to simulate learners of this type. Again, our model used the L_{RI} scheme, but with a lower value of λ, which was between 0.0142 to 0.0152.

[4]The question of how to deal with chapters of increasing complexity is dealt with elsewhere [14]. Without belaboring the point, we mention that in order for the Domain model to keep the identity of superior actions in subsequent chapters hidden from the Student, in [14] we propose to shuffle the order of the choices that are presented to the Student. Therefore, the Student can not use his prior knowledge to "short-cut" an answer to more difficult learning material.

The simulations were performed for the different 4-action and 10-action benchmark Environments, for which the threshold T was set to be 0.99, and the number of experiments (NE) = 75. The results of these simulations are described below.

5.1 Results using 4-action and 10 action Environments

The teaching Environments in the simulations have been represented by two benchmark sets of Environments. The first set includes two 4-action Environments ($E_{4,A}$ and $E_{4,B}$), and the second set contains two 10-action benchmark Environments ($E_{10,A}$ and $E_{10,B}$). The 4-action Environment represents a multiple-choice question with 4 options, whereas the 10-action Environment represents a more difficult multiple-choice question, namely with 10 options. The Students in the simulations needed to learn the responses for the questions and determine the choice that possessed the minimum penalty probability.
For $E_{4,A}$ and $E_{10,A}$, the λ of the Student Simulators LA were set to be:

- 0.0127 for the Fast learning Student.
- 0.0192 for the Normal learning Student.
- 0.0142 for the Below-Normal learning Student.

Also, for $E_{4,B}$ and $E_{10,B}$, the λ of the Student Simulators LA were:

- 0.0041 for the Fast learning Student.
- 0.0182 for the Normal learning Student.
- 0.0152 for the Below-Normal learning Student.

For the 4-action Environments, the reward probabilities were:
$E_{4,A} = \{0.7 \quad 0.5 \quad 0.3 \quad 0.2\}$ and
$E_{4,B} = \{0.1 \quad 0.45 \quad 0.84 \quad 0.76\}$.
Similarly, for the 10-action Environments, the two settings for the reward probabilities were:

$$E_{10,A} = \{0.7 \quad 0.5 \quad 0.3 \quad 0.2 \quad 0.4 \quad 0.5 \quad 0.4 \quad 0.3 \quad 0.5 \quad 0.2\} \text{ and}$$
$$E_{10,B} = \{0.1 \quad 0.45 \quad 0.84 \quad 0.76 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.7 \quad 0.5 \quad 0.3\}.$$

The simulations were performed for the different Environments and types of Students, as described above. Also, the experiments were conducted by controlling the Domain model with different factors of difficulty μ, from the range of 1.0 (no difficulty) to 0.3. The results of these simulations are provided in Table 1.

The results demonstrate that the difficulty of the Domain knowledge increased by decreasing μ for both the Normal and Below-Normal learners. As opposed to this, Fast learners were apparently not adversely affected by the increasing difficulty of the Domain knowledge. For example, in the $E_{4,A}$ Environment, a Normal learner Student LA converged in 975 iterations to learn the material in an Environment without any enhanced level of difficulty (μ=1.0), while it converged in 1,474 iterations when μ decreased to 0.6, which represents an increase of 51% in

Env.	No. of actions	Chapter	μ (diffic. factor)	No. iterations for **Fast Learner** to converge	No. iterations for **Normal Learner** to converge	No. iterations for **Below Norm. Learner** to converge
$E_{4,A}$	4	1	1.0	563	975	1,380
		2	0.9	542	1,051	1,497
		3	0.8	530	1,192	1,628
		4	0.7	515	1,321	1,722
		5	0.6	506	1,474	2,085
		6	0.5	473	1,682	2,245
		7	0.4	492	1,918	3,077
		8	0.3	523	2,393	3,512
$E_{4,B}$	4	1	1.0	1,480	2,046	2,459
		2	0.9	1,420	2,326	2,799
		3	0.8	1,388	2,247	2,894
		4	0.7	1,445	2,500	3,525
		5	0.6	1,443	2,631	3,533
		6	0.5	1,378	2,897	3,887
		7	0.4	1,445	3,569	4,652
		8	0.3	1,407	3,651	5,036
$E_{10,A}$	10	1	1.0	680	1,320	1,728
		2	0.9	684	1,442	1,972
		3	0.8	660	1,485	2,256
		4	0.7	667	1,731	2,386
		5	0.6	641	1,963	2,845
		6	0.5	650	2,226	3,424
		7	0.4	656	2,604	3,965
		8	0.3	711	3,105	5,085
$E_{10,B}$	10	1	1.0	1,631	2,286	2,773
		2	0.9	1,623	2,282	2,662
		3	0.8	1,580	2,608	3,319
		4	0.7	1,555	2,879	3,644
		5	0.6	1,590	2,928	3,580
		6	0.5	1,527	3,478	1,460
		7	0.4	1,641	3,717	4,628
		8	0.3	1,715	4,015	5,776

Reward probabilities for 4-action Environments are:

$E_{4,A}$: 0.7 0.5 0.3 0.2

$E_{4,B}$: 0.1 0.45 0.84 0.76

Reward probabilities for 10-action Environments are:

$E_{10,A}$: 0.7 0.5 0.3 0.2 0.1 0.5 0.4 0.3 0.5 0.2

$E_{10,B}$: 0.1 0.45 0.81 0.76 0.2 0.4 0.6 0.7 0.5 0.3

Table 1: Convergence of the Student Simulators learning in the benchmark Four/Ten-Action Environments by increasing the level of difficulty in the Domain knowledge.

the learning time for the Student. When the difficulty increased further by setting μ=0.3, the iterations needed for learning increased to 2,393, which represents an increase of 145% of the time required to learn when compared to the original benchmark Environment.

Similar results were also recorded for the Below-Normal learner, where he learned the material in 1,380 iteration in an Environment without any added difficulty. When the difficulty of the material increased corresponding to a value of μ=0.6, the number of iterations increased to 2,085, which represents a 51% increase in the learning time. The learning time increased to 3,512 iterations when the control parameter μ was 0.3, which represents a 154% increase in the learning time from the original benchmark.

On the other hand, the results showed that Fast learners were not adversely affected by increasing the difficulty in the Environment. Indeed, the number of iterations needed to learn the teaching material were almost constant. This seems to also be the case for *real-life* Students.

Similar results are also observed for the other Environments ($E_{4,B}$, $E_{10,A}$, and $E_{10,B}$), as seen from Table 1.

Figure 2 depicts graphically the results of the simulations. For each benchmark Environment, it displays the relationship between the number of iterations and μ. The reader should observe the apparent "proportional" increase in the number of iterations by increasing the complexity (i.e. by decreasing μ) for both Normal and Below-Normal learners. It also shows that Fast learners were not affected by the increased complexity of the problem.

6 Conclusion

This paper introduced a novel approach of modeling the Domain knowledge in a Tutorial-*like* system. In this model, the Domain knowledge is presented in different chapters, where the difficulty of the learned knowledge increased with the subsequent chapters.

The Domain knowledge has been modeled using the concept of Environments in a LA paradigm, from which the Student Simulator LA are trying to learn. The model presented in the paper showed that the difficulty of the Domain knowledge (as increasingly more complex chapters were encountered) could be increased by decreasing the range of the penalty probabilities of all the pertinent actions by multiplying them with a factor, μ. The main results that we have obtained is that the learning time increased for Normal and Below-Normal learners as the difficulty of the Domain knowledge increased. This was not the case for Fast learners, which seems to be consistent with our experience with *real-life* Students.

The Teacher will be using the Domain model to present the teaching material in a chapter-wise fashion to the Students. He will need to determine when the chapter complexity can be increased, and how prior knowledge can be used by the Student LA in such Tutorial-*like* systems. This is currently being done elsewhere [14].

For future work, we are considering how we can use this approach to model

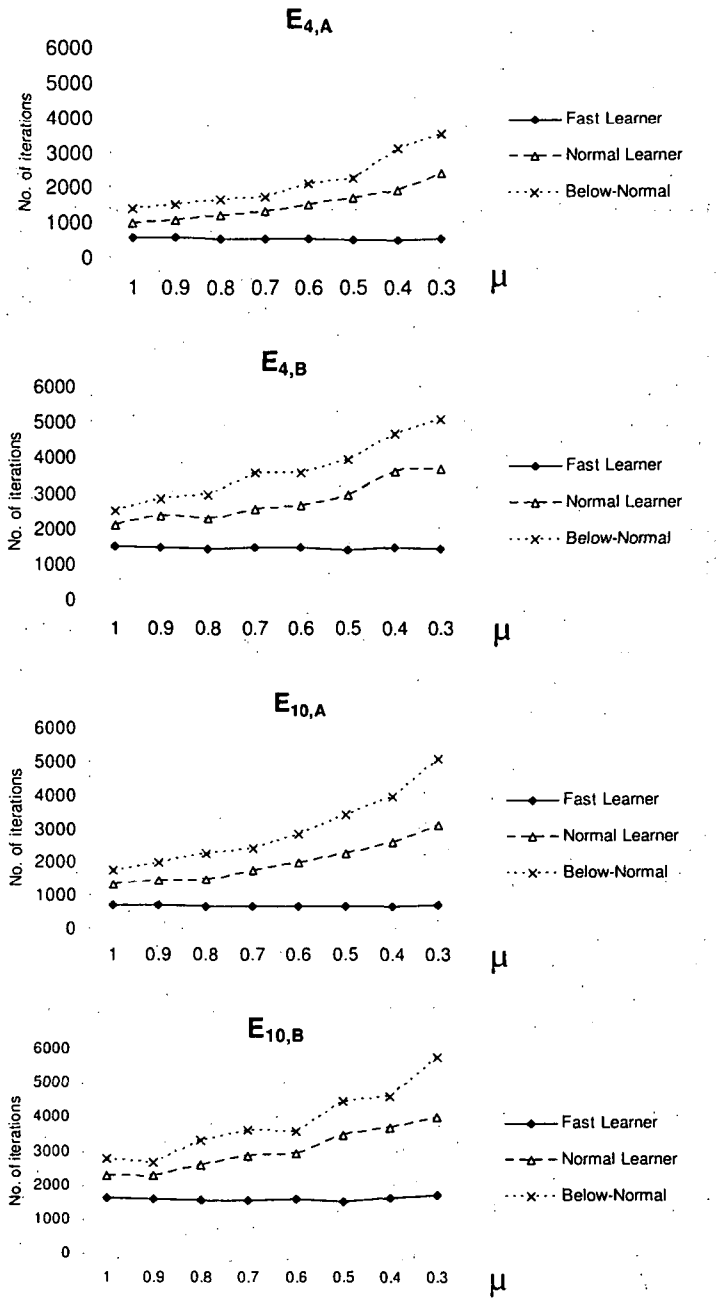

Figure 2: The effect of increasing the difficulty of the Domain model on the different types of Students in the four benchmark Environments.

the increasing complexity of *real-life* domain knowledge, where the multiple choice questions that the Student answers are from a real domain. A more distant future work is to port this approach to be used in traditional Tutorial systems, where the Students can be taught uncertain domain knowledge.

References

[1] Agache, M. and Oommen, B. J. Generalized pursuit learning schemes: New families of continuous and discretized learning automata. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):738–749, December 2002.

[2] Atlassis, A. F., Loukas, N. H., and Vasilakos, A. V. The use of learning algorithms in atm networks call admission control problem: A methodology. *Computer Networks*, 34:341–353, 2000.

[3] Atlassis, A. F. and Vasilakos, A. V. The use of reinforcement learning algorithms in traffic control of high speed networks. *Advances in Computational Intelligence and Learning*, pages 353–369, 2002.

[4] Atolagbe, T. A. and Hlupic, V. SimTutor: A multimedia intelligent tutoring system for simulation modeling. In Andraddttir, S., Healy, K. J., Withers, D. H., and Nelson, B. L., editors, *Proceedings of the 29th Conference on Winter Simulation*, pages 504–509, Atlanta, Georgia, 1997.

[5] Baddeley, A. and Turner, R. Spatstat: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12:1–42, 2005.

[6] Barzohar, M. and Cooper, D. B. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:707–722, 1996.

[7] Bertsimas, D. J. and Ryzin, G. Van. Stochastic and Dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, 41:60–76, 1993.

[8] Bettstetter, C., Hartenstein, H., and Pérez-Costa, X. Stochastic properties of the random waypoint mobility model. *Journal Wireless Networks*, 10:555–567, 2004.

[9] Brandeau, M. L. and Chiu, S. S. An overview of representative problems in Location Research. *Management Science*, 35:645–674, 1989.

[10] Collins, J. J., Chow, C. C., and Imhoff, T. T. Aperiodic stochastic resonance in excitable systems. *Physical Review E*, 52:R3321–R3324, 1995.

[11] Cook, R. L. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5:51–72, 1986.

[12] Cusumano, J. P. and Kimble, B. W. A stochastic interrogation method for experimental measurements of global dynamics and basin evolution: Application to a two-well oscillator. *Nonlinear Dynamics*, 8:213–235, 1995.

[13] Fischetti, E. and Gisolfi, A. From computer-aided instruction to intelligent tutoring systems. *Educational Technology*, 30(8):7–17, 1990.

[14] Hashem, M. K. *Learning Automata Based Intelligent Tutorial-like Systems.* PhD thesis, School of Computer Science, Carleton University, Ottawa, Canada, 2007.

[15] Hashem, M. K. and Oommen, B. J. On using learning automata to model a student's behavior in a tutorial-*like* system. In *Proceedings of the IEA/AIE 2007: The 20th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems*, pages 813–822, Kyoto, Japan, June 2007.

[16] Kabudian, J., Meybodi, M. R., and Homayounpour, M. M. Applying continuous action reinforcement learning automata (CARLA) to global training of hidden markov models. In *Proceedings of the International Conference on Information Technology: Coding and Computing , ITCC'04*, pages 638–642, Las Vegas, Nevada, 2004.

[17] Lakshmivarahan, S. *Learning Algorithms Theory and Applications.* Springer-Verlag, 1981.

[18] Meybodi, M. R. and Beigy, H. New learning automata based algorithms for adaptation of backpropagation algorithm pararmeters. *International Journal of Neural Systems*, 12:45–67, 2002.

[19] Misra, S. and Oommen, B. J. GPSPA: A new adaptive algorithm for maintaining shortest path routing trees in stochastic networks. *International Journal of Communication Systems*, 17:963–984, 2004.

[20] Najim, K. and Poznyak, A. S. *Learning Automata: Theory and Applications.* Pergamon Press, Oxford, 1994.

[21] Narendra, K. S. and Thathachar, M. A. L. *Learning Automata: An Introduction.* Prentice-Hall, New Jersey, 1989.

[22] Obaidat, M. S., Papadimitriou, G. I., Pomportsis, A. S., and Laskaridis, H. S. Learning automata-based bus arbitration for shared-medium ATM switches. *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 32:815–820, 2002.

[23] Obaidat, M. S., Papadimitrious, G. I., and Pomportsis, A. S. Learning automata: Theory, paradigms, and applications. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):706–709, December 2002.

[24] Omar, N. and Leite, A. S. The learning process mediated by intelligent tutoring systems and conceptual learning. In *International Conference On Engineering Education*, page 20, Rio de Janeiro, 1998.

[25] Oommen, B. J. and Agache, M. Continuous and discretized pursuit learning schemes: Various algorithms and their comparison. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 31:277–287, 2001.

[26] Oommen, B. J. and Croix, E.V. de St. Graph partitioning using learning automata. *IEEE Transactions on Computers*, C-45:195–208, 1995.

[27] Oommen, B. J. and Roberts, T. D. Continuous learning automata solutions to the capacity assignment problem. *IEEE Transactions on Computers*, C-49:608–620, 2000.

[28] Paola, M. Digital simulation of wind field velocity. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76:91–109, 1998.

[29] Papadimitriou, G. I. and Pomportsis, A. S. Learning-automata-based TDMA protocols for broadcast communication systems with bursty traffic. *IEEE Communication Letters*, pages 107–109, 2000.

[30] Poznyak, A. S. and Najim, K. *Learning Automata and Stochastic Optimization.* Springer-Verlag, Berlin, 1997.

[31] Rowlingson, B. S. and Diggle, P. J. *SPLANCS: spatial point pattern analysis code in S-Plus.* University of Lancaster, North West Regional Research Laboratory, 1991.

[32] Sanders, J. R. In *Twenty-fourth Annual Meeting of The Joint Committee on Standards for Educational Evaluation*, October 1998.

[33] Self, J. The defining characteristics of intelligent tutoring systems research: ITSs care, precisely. *International Journal of AI in Education*, 10:350–364, 1999.

[34] Seredynski, F. Distributed scheduling using simple learning machines. *European Journal of Operational Research*, 107:401–413, 1998.

[35] Thathachar, M. A. L. and Oommen, B. J. Discretized reward-inaction learning automata. *Journal of Cybernetics and Information Science*, pages 24–29, Spring 1979.

[36] Thathachar, M. A. L. and Sastry, P. S. Varieties of learning automata: An overview. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):711–722, December 2002.

[37] Thathachar, M. A. L. and Sastry, P. S. *Networks of Learning Automata: Techniques for Online Stochastic Optimization.* Kluwer Academic, Boston, 2003.

[38] Tsetlin, M. L. *Automaton Theory and the Modeling of Biological Systems.* Academic Press, New York, 1973.

[39] Unsal, C., Kachroo, P., and Bay, J. S. Simulation study of multiple intelligent vehicle control using stochastic learning automata. *Transactions of the Society for Computer Simulation International*, 14:193–210, 1997.

[40] Vasilakos, A., Saltouros, M. P., Atlassis, A. F., and Pedrycz, W. Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques. *IEEE Transactions on Systems Science, and Cybernetics, Part C*, 33:297–312, August 2003.

[41] Vasilakos, A. V., Saltouros, M. P., Atlassis, A. F., and Pedrycz, W. Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques. *IEEE Transactions on Systems, Man, and Cybernetics: Part C*, 33:297–312, 2003.

[42] Winkels, R. and Breuker, J. What's in an ITS? a functional decomposition. In Costa, E., editor, *New Directions for Intelligent Tutoring Systems.* Spring-Verlag, Berlin, 1990.

Received 18th July 2008

Acta Cybernetica **19** (2010) 655–660.

Automata Depository Model with Autonomous Robots*

Zoltán Szabó† Balázs Lájer† and Ágnes Werner-Stark‡

Abstract

One of the actual topics on robotis research in the recent decades is the robots' autonomy. The methods of self-sufficient problem-solving of the machines brings on several questions in programming, so mobile robots started to extend as tools of education as well.

Our final goal was in this project to create the model of an automata depository that constitutes a closed system from the users' point of view. We model such circumstances that make autonomy important like extreme high or low temperature, closeness of dangerous materials.These circumstances substantiates the need of robots and that they have to solve their problems self-sufficiently, without any direct human interaction.

The model builds up from two main components: the Central Controlling Unit (CCU), and the group of robots. The robots ply in the depository using the line following method. During their activity may turn up some conflict situations, whose autonomous handling is the main topic of our research. Using the right wayfinder algorithm and the representation of the map of the depository, the robots find out after a short information excange, who of them has to give way to the other in order to solve the conflict in optimal time.

The communication between the LEGO MINDSTORMS NXT Robots and the Central Controlling Unit is based on a BlueTooth connection.

The robots' autonomy means that if they loose connection with the CCU, they can finish their commands that they have already received. Nevertheless navigating their physical relocation and sense any incidental new barrier is absolutely their task.

Keywords: autonomous robot, mindstorms nxt, solving conflict situation

1 Introduction

This project has been created for a Scientific Student Conference. Our final goal was in this project to create the model of an automata depository that constitutes a closed system from the users' point of view. We model such circumstances

*This work was supported by the LEGO Hungária Ltd.

†E-mail: {szabo.zoltan,lajer.balazs}@door2world.hu, werner@virt.uni-pannon.hu

that make autonomy important like extreme high or low temperature, closeness of dangerous materials. These circumstances substantiates the need of robots and that they have to solve their problems self-sufficiently, without any direct human interaction.

The system consists of two parts: the Central Controlling Unit (CCU) and the group of robots. The user shall not do any interaction with the robots, only with the CCU through the user interface. The CCU and the robots bring the user's queries into effect independently, so we can mention the systems autonomy as well.

First of all we had to find a proper hardware for the project. After having examined some robot kits that can be found on the internet, we chose the Lego Mindstorms NXT set, as the robot can be built up easily and people can start work on the software before long. We decided to use the Lejos NXJ platform what is a little JAVA Virtual Machine running on the NXT Bricks, so we could develop our program in object-oriented method.

2 Build-up and software of the robots

Using the sensors that can be found in the Mindtorms NXT set, we could easily build up a robot that we could use for this research. The robots stand on a three-wheel frame that makes them stable enough. We used only two of the available sensors: the light sensor for the line-following and the ultrasonic sensor on the top of the robot in order to sense if it reaches a barrier and to know if the robots fork-lift is fully under the box it is going to lift.

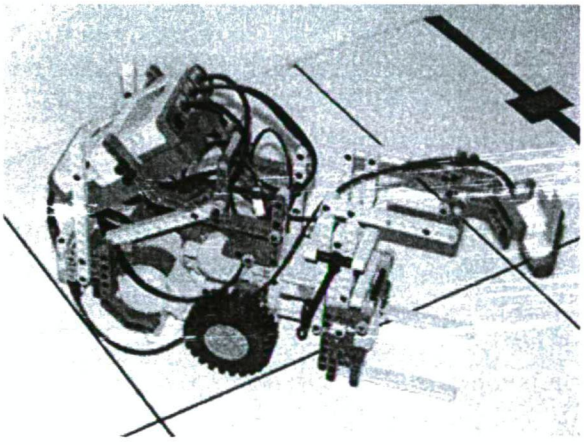

Figure 1: Top view

After having implemented some basic functionalities such as moving forward to a given distance, turning in a given degree, getting to a given point in a virtual frame of reference or lifting up and letting down the fork-lift, we started building their main software. Its tasks are: to process a route computed by the Central

Controller Unit, to monitor every important information to the CCU, discover new barriers, to keep in touch with the other robots. Our first idea was to implement a recursive routeplanner algorithm that runs on the NXT Brick, but unfortunately we had to face the fact that the recursion uses all the available memory after the second step. Because of we had no time and close deadlines with this project, we decided to use the CCU to plan the route. This algorithm is a breadth-first search algorithm. The adjacency list stores the path points on the map. By this way we use every time one of the shortest routes. There is a work in progress to develop an other non-recursive algorithm that can run on the NXT Bricks.

The base class of the robots' software is the Controller class. The tasks mentioned above are shared between two threads: one thread is responsible for the navigation and all the other functionalities that help the robot to complete its assignment, the other thread makes the communication.

3 Navigation

The virtual map of the depository is a frame of reference with synchronizing points in the middle of each field. We had some trouble because of the imprecision of the lego parts, that's why we chose the line-following method. Because of the same reason, we had to take care to implement some corrections when turning the robots. If the robot turns 90 degrees in a direction during processing its route, the result was not always correct. The best solution was that if the robot could not sense the line after having turned, it starts iteratively swivel in both directions by increasing angles. This way it will surely find the line. The robots orientate themselves by keeping their last position in their memory. They know that they have reached a synchronizing point by sensing again black line after they had left it before. If they cannot sense black again, then they have left the line because of some reason, so they have to search for it again using the correction method mentioned above.

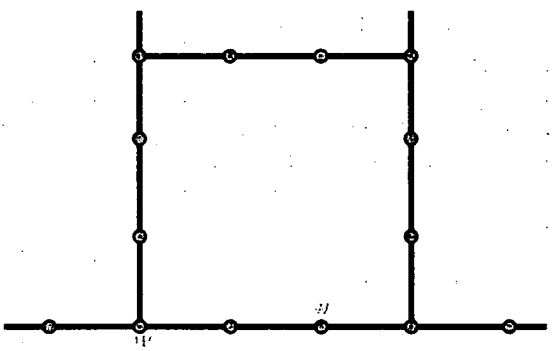

Figure 2: The map

4 Conflict Situations

In our model we determined three types of conflict situations. One is that two robots are on their way, and their routes cross each other. The second one is when only one of them has an actual task (Robot A), the other is idle (Robot B) but staying on the route that Robot A has to process. In the third situation both robots have their task, but one of them could go faster on its route, so it comes up with the other. I this case the conflict handling is very simple, the faster robot only have to wait for the other.

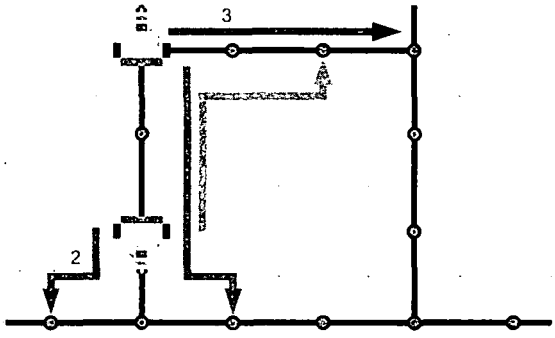

Figure 3: An example conflict situation

In every monitoring message the robots send the actual coordinates on the map. They send these information for each other not just the for the CCU, but because of some issues with the firmware these data go through a pipe in the CCU (it will be explain below). By this way they can calculate the distance from the other. They use the Pythagorean theorem, but we are working on an other more effective method which will use the map for calculating.

When they got the distance they have to determine whether this is a conflict situation and if it is what kind of conflict situation.

First of all they examine the distance to know if it is under the threshold or not. If it is over the threshold they can go on. If it is under or equal with the threshold they have to use one of the avoiding tactics. To determine the avoiding case they get more information. For first the state of the other robot (idle, busy), and the route of the other. They can determine wherher their routes have common point or not. If there is at least one common point then they have to continue the method else they can go on their own route. If they know that their routes cross each other then they search the first free path point and send the length of the avoiding route to the other robot. A free path point is one of the path points on the map along the other's route but not on it. The two robots can decide which has to do avoiding, it depends on who has the shortest route. After the avoiding robot is out of the way, sends a signal to the other to continue its own route. When the robot finished provessing the route, sends a resume signal to the avoiding robot to

continue the interrupted task. In this case the avoiding robot go to the coordinate where the task was interrupted, and continues it.

5 System architecture and software

The CCU works as a bridge between the TCP/IP network and the Bluetooth network. It accepts connections from the users via TCP/IP protocol. The users can use a string-based protocol to order actions in the system. We are working on a graphical user interface to ease the users' work.

On the other side the CCU has to build and keep persistent the connection with the robots. Because the CCU initiates the connections, it has to know the name and the address of the Bluetooth devices. The core of the CCU is the ServerCore class. This is the main controller which is responsible for delegating tasks to the robots and serving information to the users. It starts two threads to listen to the user connections and for the robot communication control. These are the channels for the communication between the two networks.

The robot's software has the Controller main class which is responsible for starting the communication thread and for initialising the navigator classes and reset the position of the fork-lift.

The solution of the problem with the firmware Bluetooth implementation is in the communication thread. We had to face the fact that in this version of the firmware we can use only one connection in one time. So we can not communicate with the CCU and the other robots at the same time as we planned before. So we modified the protocol of the CCU-Robot communication and made it able to let through the messages between the robots. We use an interface in these classes to make it possible to modify the program when multiple connections are available.

6 Problems and future plans

Some minor problems turn up because of the sensitivity and imprecision of the sensors. We draw the inference that the light sensor need constant circumstances after its thresholds had been set. It measured false data when the sun shone through the window or when the lights were on. It was very difficult to find the right values for the ultrasonic sensor and the right form for the box in the depository to make them work together.

Our main problem was the imperfection of the BlueTooth connectivity in the NXJ platform. After the connection between a robot and the CCU has been established, we tried to command the robot to build up a connection with the another. Then the robot's software suddenly froze, and we did not know what happened. Having searched for the implementation of the BlueTooth connectivity in the NXJ code, we found that only one listener can be active in a robot. While we were searching for some solution on the internet, we could see that several other projects also missed this feature so we look forward for the next version of the NXJ firmware.

7 Conclusion

We succeed to create a model of an automata depository in which the user only needs to command a computer, there is no interaction needed between human and robot. The CCU and the robots can manage to bring into effect the changes that the user had queried. We found the Lego Mindstorms NXT robots very useful in teaching programming, especially robotics and embedded systems.

References

[1] Druin, A., Hendler, J. Robots for kids: Exploring new technologies for learning. San Diego, CA: Academic Press, 2000.

[2] Wie hauche ich dem kleinen Roboter "Asuro" das Leben ein? http://www.dlr.de/schoollab/desktopdefault.aspx/tabid-2980/4537_read-6668/

[3] LEGO, LEGO MINDSTORM set for Schools # 9790. Billund, Denmark: The LEGO Group, 1999.

[4] http://claraty.jpl.nasa.gov/man/overview/index.php

[5] Lui, M., Hsiao, Y. Middle school students as multimedia designers: A project-based learning approach. Journal of Interactive Learning Research, 13(4), 311-337, 2002.

[6] Java for LEGO Mindstorms. http://lejos.sourceforge.net/p_technologies/nxt/nxj/api/index.html

[7] Cormen, Rivest, Leiserson.. Algoritmusok. 2003.

[8] http://mindstorms.lego.com/eng/Overview/Bluetooth.aspx

[9] http://www.daimi.au.dk/~dam/logs/doku.php?id=projekt2#thursday_d._13_12-07

Received 3rd January 2009

Acta Cybernetica **19** (2010) 661–672.

An Algorithmic Comparison of Three Scientific Impact Indices*

Gerhard J. Woeginger†

Abstract

We use tools from Theoretical Computer Science to analyze the computational complexity of determining the h-index, the g-index, and the w-index in various models of computation. Our results confirm the natural intuition that the h-index is an easier concept than the g-index, which in turn is an easier concept than the w-index.

Keywords: scientific impact measure, efficient algorithm, computational complexity

1 Introduction

Citation analysis as a method for ranking scientific journals, publications, and researchers is an old idea that goes at least back to Gross and Gross ([7]). A simple and natural approach for quantifying the scientific productivity and scientific impact of a researcher with $n \geq 0$ publications is based on the so-called *citation sequence* $\langle x_1, \ldots, x_n \rangle$ of the researcher; here the kth element x_k states the total number of citations to the kth publication. A *scientific impact index* assigns to every such citation sequence a corresponding non-negative integer that concisely expresses the productivity, quality, and visibility of this researcher.

In recent years, the h-index of Jorge Hirsch ([8]) and the g-index of Leo Egghe ([5], [6]) have become particulary popular impact indices in this area. Woeginger ([11],[12]) performed an axiomatic analysis of the h-index and the g-index, and as a by-product these axiomatic investigations lead to the definition of the so-called w-index.

The h-index: A scientist has index h, if h is the largest integer such that at least h of his articles have received at least h citations each.

*This work has been supported by the Netherlands Organisation for Scientific Research (NWO), grant 639.033.403, and by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society)

†Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands., E-mail: gwoegi@win.tue.nl

The g-index: A scientist has index g, if g is the largest integer such that his top g articles have received together at least g^2 citations.

The w-index: A scientist has index w, if w is the largest integer such that w of his articles have received at least $1, 2, 3, 4, \ldots, w$ citations, respectively.

In this paper, we will discuss the algorithmic complexity of computing the h-index, the g-index, and the w-index of a researcher, and we will thereby compare the relative effort needed for determining these three indices. Let us start with an example, and let us consider a researcher X whose 19 publications have attracted the following numbers of citations:

$$10,\ 20,\ 20,\ 12,\ 55,\ 2,\ 2,\ 4,\ 0,\ 14,\ 15,\ 14,\ 0,\ 0,\ 9,\ 10,\ 2,\ 1,\ 3.$$

It is not difficult to see that the h-index of Mr. X is 9 (hint: search for 9 citation numbers that are all at least 9). The task becomes much easier for the human eye, if one first brings the numbers into non-increasing order as follows:

$$55,\ 20,\ 20,\ 15,\ 14,\ 14,\ 12,\ 10,\ 10,\ 9,\ 4,\ 3,\ 2,\ 2,\ 2,\ 1,\ 0,\ 0,\ 0.$$

What about the g-index of Mr. X? Since the sum of the 13 highest numbers in this sequence is $188 \geq 13^2$, and since the sum of the 14 highest numbers is $190 < 14^2$, we see that his g-index is 13. Finally the w-index of this researcher is 14, since the top publication has $55 \geq 14$ citations, since the second-strongest publication has $20 \geq 13$ citations, the third-strongest $20 \geq 12$, the fourth-strongest $15 \geq 11$, and so on down to the publication at rank 14 with $2 \geq 1$ citations.

After building up some more experience and after determining these three impact indices for several dozens of researchers, one inevitably comes to the following conclusion: The h-index is the most primitive index among the three, and usually can be found quickly and easily. The g-index takes somewhat more effort to compute, and in particular involves the summing of a lot of numbers. Finally the w-index seems to be a pain in the neck, and one really has to compare a whole lot of numbers one by one for finding the w-index.

Contributions of this paper. The goal of this paper is to confirm the fuzzy observations that were claimed in the preceding paragraph without much justification. We will reach this goal by applying machinery from algorithms theory, and by analyzing the time complexity for computing the three impact indices under various models of computation: First, we will consider the model where the citation sequence is stored in fast random access memory and has already been sorted into non-increasing order. Secondly, we consider the situation where the citation sequence is in random access memory, but is unordered. Thirdly, we will consider the model where the citation sequence is stored in sequential memory (as for instance on a tape), and where accessing a data element cannot be done instantaneously, but costs some data reading and data processing time.

It turns out that in all three models the g-index can be computed within a time complexity that is proportional to the length n of the citation sequence. The

h-index behaves similarly as the g-index, but in the first of the above models it is (provably!) much easier to compute. The w-index can also be computed with linear effort in the first and second model, but in the third model it is (provably!) harder to compute than the h-index and the g-index. Our results are summarized in Table 1.

We stress that our main contribution does not consist of the algorithms derived in this papers: These algorithms are purely theoretical constructions that have been tailored to work under certain simplified models of computation and that most probably have no practical relevance. In fact the calculation of all three indices is relatively easy and could be done quickly even for millions of data records on any modern PC by the most primitive and direct implementations. Our main contribution is purely conceptual: We use these tools from Theoretical Computer Science to provide mathematical evidence that the h-index is a more primitive concept than the g-index and that the g-index is a more primitive concept than the w-index. And actually, we are not aware of any other scientific tools that would be able to yield such a result.

The paper is organized as follows. Section 2 gives a soft introduction into the analysis of algorithms, and also specifies the three considered models of computation more precisely. Sections 3, 4, 5 respectively discuss how to compute the considered impact indices in the three models of computation. Section 6 gives the conclusion.

Table 1: Asymptotic worst case time complexities for computing various impact indices (of citation sequences with n elements) in various models of computation. All nine results are asymptotically best possible.

	h-index	g-index	w-index
Sorted data in random access memory	$\log n$	n	n
Unsorted data in random access memory	n	n	n
Unsorted data in sequential memory	n	n	$n \log n$

2 Preliminaries on algorithms and computation

In this section we summarize some very basic facts on the analysis of algorithms and on models of computation. For more information on these concepts, we refer the reader to the books of Aho, Hopcroft, and Ullman ([1]), Cormen, Leiserson, and Rivest ([4]), and Papadimitriou ([10]).

Algorithms can be evaluated by a variety of criteria. The most common criterion is the growth of the running time required to solve larger and larger instances of a problem: Every instance of a problem has a certain *size*, which measures the quantity of input data. For example the size of a citation sequence $\langle x_1, \ldots, x_n \rangle$ is the number n of elements in the sequence. The *time complexity* of an algorithm is a worst case measure and denotes the maximum number of elementary steps needed by the algorithm as a function of the input size; in other words if an algorithm has time complexity $T(n)$ then it can solve all instances of size n within $T(n)$ elementary steps. The *asymptotic time complexity* of an algorithm is the limiting behavior of the worst case time complexity as size increases. If an algorithm can process all inputs of size n in time $T(n) \leq cn^2$ for some constant c, then we say that its time complexity is $O(n^2)$. More precisely, a time complexity function $f(n)$ is said to be in $O(g(n))$, if there exists a constant c such that $f(n) \leq cg(n)$ holds for all positive n.

In the preceding paragraph we have specified the running time of an algorithm as the number of elementary steps. The definition of an elementary step heavily depends on the underlying model of computation. The most common model of computation is the random access machine (RAM) model. Every storage cell of a RAM can hold an integer. The integers in two storage cells can be added, subtracted, multiplied, divided, or compared against each other in one elementary step. The contents of every storage cell can be accessed instantaneously, and the results of additions, subtractions, multiplications, and divisions can be stored instantaneously into new storage cells. If a citation sequence $\langle x_1, \ldots, x_n \rangle$ is stored in the cells of a RAM, then each element x_k can be accessed through its index k within a single elementary step. The random access machine is the underlying model of computation for Sections 3 and 4: In Section 3 we will additionally assume that the input citation sequence has already been sorted into non-increasing order $x_1 \geq x_2 \geq \cdots \geq x_n$. In Section 4 we discuss the case of unordered citation sequences.

Another fundamental model of computation stores the data in *sequential memory*. Also in this model every storage cell can hold an integer, but now accessing the storage cells is more expensive: The data is stored and processed on a fixed number of tapes (or lists), and the data on every tape is accessed through a read-write head for this tape. In this model the contents of a storage cell can not be accessed instantaneously: For instance, if we want to access the contents of storage cell #1 and afterwards the contents of storage cell #1.000, then the read-write head must inbetween move from cell #1 to cell #1.000, which takes 999 elementary steps. Hence moving one cell along the tape, reading a cell, and writing into a cell form elementary steps in the sequential memory model. The model also assumes that there are a small fixed number of register cells (in fast memory) available that can be used for temporarily storing data. Addition, subtraction, multiplication, division, and comparison of the integers in the register cells are elementary steps.

In Section 5 we will discuss the computation of impact indices under the sequential memory model. As a main tool we will use a celebrated algorithm for selecting the m-largest element among n numbers.

Proposition 1. *(Blum, Floyd, Pratt, Rivest, and Tarjan, [3])*
Consider n numbers that are stored in random access memory or in sequential memory. Then the m-largest element among these n numbers can be determined in linear time $O(n)$.

3 The case of sorted data in random access memory

Throughout this section we consider citation sequences $\langle x_1, \ldots, x_n \rangle$ in which the elements are in non-increasing order $x_1 \geq x_2 \geq \cdots \geq x_n$. We assume that the data is stored in fast random access memory, so that each element x_k can be accessed immediately through its index k.

We start our investigations with the h-index. In this case our main tool is the classical *Binary Search* procedure, as discussed for instance in the textbook of Aho, Hopcroft, and Ullman ([1]). We will apply Binary Search to the following auxiliary problem: Given a sorted sequence $y_1 \geq y_2 \geq \cdots \geq y_n$ of integers with $y_1 \geq 0$, determine the largest index h for which y_h is non-negative. The main idea of Binary Search is to narrow down the search space to smaller and smaller intervals $[\ell, u]$. In the beginning the search space is the entire interval $[1, n]$ so that $\ell = 1$ and $u = n$. Then Binary Search looks at the value of the middle element y_m with $m := \lfloor (\ell + u)/2 \rfloor$. If y_m is negative, then the new search space becomes $[1, m-1]$. If y_m is non-negative, then the new search space becomes $[m, n]$. This is repeated until the search space has been narrowed down to at most two elements, which are then checked separately. Since every search step removes half of the search interval, the time complexity $T(n)$ satisfies

$$T(n) \;\leq\; T(\lceil n/2 \rceil) + c,$$

where c is the time needed for computing m and for querying the mth element. Now an easy induction yields $T(n) \leq d\lceil \log_2 n \rceil$ for some appropriate constant d.

Theorem 1. *The h-index of a sorted citation sequence $x_1 \geq x_2 \geq \cdots \geq x_n$ can be determined in $O(\log_2 n)$ time. No algorithm can have a worst case time complexity that is asymptotically better than $O(\log_2 n)$.*

Proof. The h-index of a sequence $x_1 \geq x_2 \geq \cdots \geq x_n$ is the largest integer h for which the value $y_h = x_h - h$ is non-negative. Hence the h-index can be computed by solving the auxiliary problem discussed above for the auxiliary sequence defined by $y_i = x_i - i$ for $1 \leq i \leq n$. This yields the $O(\log_2 n)$ time complexity claimed in the positive part of the theorem.

For the negative part we use an information-theoretic argument. For $1 \leq k \leq n$ consider the sorted citation sequence $S_{n,k}$ that consists of k elements of value n followed by $n - k$ elements of value 0. Clearly the h-index of $S_{n,k}$ equals k. Now

consider an arbitrary algorithm A for computing the h-index. Whenever A queries one element in such a sequence $S_{n,k}$, it only gains a single bit of information: The queried element is either equal to n, or it is not (in which case it is 0). Since the algorithm needs at least $\log_2 n$ bits to distinguish between the n possible outcomes, it must query $\log_2 n$ elements in the worst case. □

Theorem 2. *The g-index of a sorted citation sequence $x_1 \geq x_2 \geq \cdots \geq x_n$ can be determined in $O(n)$ time. No algorithm can have a worst case time complexity that is asymptotically better than $O(n)$.*

Proof. First we compute the sum $s[k] = \sum_{i=1}^{k} x_i$ for $k = 1, \ldots, n$. This can be done in overall linear time $O(n)$, since $s[1] = x_1$ and since $s[k] = s[k-1] + x_k$ holds for $k = 2, \ldots, n$. The g-index is then the largest index k with $s[k] \geq k^2$. This yields the $O(n)$ time algorithm for the positive part of the theorem.

For the negative part we consider the following sorted citation sequence S'_n: The first element of sequence S'_n is $\frac{1}{2}(n^2+n)$. The remaining $n-1$ elements in S'_n are the values $1, 2, 3, \ldots, n-1$ in decreasing order. Since the sum of all elements in sequence S'_n is n^2, its g-index is n. Furthermore for $1 \leq k \leq n$ we define a sequence $S'_{n,k}$ that results from sequence S'_n by decreasing the kth element by 1. The resulting sequence $S'_{n,k}$ is still sorted, but its g-index has dropped down to $n-1$.

Now consider an arbitrary algorithm A for computing the g-index, and feed the input sequence S'_n into algorithm A. We claim that A must inspect all n elements of sequence S'_n: If the algorithm does not inspect the kth element, then it could not distinguish sequence S'_n (with g-index n) from sequence $S'_{n,k}$ (with g-index $n-1$). □

Theorem 3. *The w-index of a sorted citation sequence $x_1 \geq x_2 \geq \cdots \geq x_n$ can be determined in $O(n)$ time. No algorithm can have a worst case time complexity that is asymptotically better than $O(n)$.*

Proof. The w-index of the sequence $x_1 \geq x_2 \geq \cdots \geq x_n$ is the largest integer k such that $x_k \geq 1$ and $x_{k-1} \geq 2$ and $x_{k-2} \geq 3$ and so on down to $x_1 \geq k$. Equivalently, we may write the w-index as

$$\begin{aligned} w &= \max\{k:\ x_m \geq k-m+1 \text{ holds for } m = 1, \ldots, k\} \\ &= \max\{k:\ x_m \geq k-m+1 \text{ holds for } m = 1, \ldots, n\} \\ &= \arg\min\{x_m + m - 1:\ 1 \leq m \leq n\}. \end{aligned}$$

The minimum value of $x_m + m - 1$ over the domain $1 \leq m \leq n$ can be determined by a single pass over the citation sequence. This yields the desired $O(n)$ time algorithm.

For the negative part we once again consider the sorted sequences S'_n and $S'_{n,k}$ that have been introduced in the proof of Theorem 2. We note that the w-index of sequence S'_n is n, whereas the w-index of every sequence $S'_{n,k}$ with $2 \leq k \leq n$ is $n-1$. Similarly as in the proof of Theorem 2 we argue that any algorithm

for computing the w-index must inspect a linear number of elements in sequence S'_n. □

4 The case of unordered data in random access memory

Throughout this section we consider unordered citation sequences $\langle x_1, \ldots, x_n \rangle$ that are stored in fast memory. As a first step we apply the following algorithm to this unordered sequence. The algorithm essentially emulates sorting through counting; see also Cormen, Leiserson, and Rivest ([4]).

Phase 1: Initialize a data array $C[0, \ldots, n]$ by setting $C[k] := 0$ for $k = 0, \ldots, n$.
Initialize a variable BIGSUM:= 0.

Phase 2: Work through the elements of the citation sequence for $k = 1, \ldots, n$.
If $0 \leq x_k < n$ holds, then set $C[x_k] := C[x_k] + 1$.
If $x_k \geq n$ holds, then set $C[n] := C[n] + 1$ and BIGSUM:=BIGSUM+x_k.

Phase 3: Output the following sorted citation sequence:
If $C[n] > 0$ holds, then output an element of value BIGSUM$-(C[n]-1)n$, followed by $C[n]-1$ elements of value n. Furthermore for $k = n-1, n-2, \ldots, 0$ output $C[k]$ elements of value k.

What does this algorithm do to a sequence $\langle x_1, \ldots, x_n \rangle$? Let us first explain the meaning of the variables: The array element $C[k]$ with $0 \leq k \leq n-1$ counts the number of elements of value k in the sequence. The last array element $C[n]$ counts the number of elements of value at least n in the sequence; these elements are called *big* elements. Finally, the variable BIGSUM contains the total size of all big elements in the sequence.

The values of the counters and of BIGSUM are determined in Phase 1 and Phase 2. In Phase 3 the citation sequence is output again, but this time in non-increasing order and with some slight changes in the values of the big elements: The total size BIGSUM of the big elements remains unchanged, but now only a single big element can be strictly larger than n. The reader may want to verify that for $n = 5$ the input sequence $\langle 0, 10, 4, 2, 20 \rangle$ will be transformed into the output sequence $\langle 25, 5, 4, 2, 0 \rangle$.

Lemma 1. *Let $x = \langle x_1, \ldots, x_n \rangle$ be an input sequence for the above algorithm, and let $y = \langle y_1, \ldots, y_n \rangle$ be the corresponding sorted output sequence. Then sequences x and y have the same h-index, the same g-index, and the same w-index.* □

Theorem 4. *If an unordered citation sequence $\langle x_1, \ldots, x_n \rangle$ with n elements is stored in random access memory, then*

(a) its h-index,

(b) its g-index, and

(c) its w-index

can all be determined in linear time $O(n)$. Under the random access model of computation, no algorithm for these three indices can have a worst case time complexity that is asymptotically better than $O(n)$.

Proof. The above algorithm takes an arbitrary input sequence, and transforms it in linear time into a sorted output sequence. By Lemma 1 the sorted sequence has the same h-index (respectively g-index and w-index) as the output sequence. Hence we may apply the fast algorithms for sorted data from the preceding section to the output sequence. This yields the positive part of the theorem.

For the negative part, we consider a citation sequence that consists of $n-1$ elements of value 0 and a single element of value 1 (that is maliciously hidden somewhere between the other $n-1$ elements). Note that the h-index, g-index, and w-index of this sequence are 1. However if an algorithm fails to inspect the element of value 1, then is cannot distinguish the sequence from the all-zero sequence whose h-index, g-index, and w-index are 0. Hence in the worst case the algorithm must query all n elements. □

5 The case of unordered data in sequential memory

Throughout this section we consider unordered citation sequences $\langle x_1, \ldots, x_n \rangle$ whose elements are stored in sequential memory.

We start our discussion with the h-index. We consider the following (purely technically motivated) generalization of the h-index that is built around an integer parameter $p \geq 0$: The $h(p)$-index of a citation sequence $\langle x_1, \ldots, x_n \rangle$ is the largest integer h such that the sequence contains at least h elements that all have value at least $h+p$.

Lemma 2. *Let $y = \langle y_1, \ldots, y_r \rangle$ and $z = \langle z_1, \ldots, z_s \rangle$ be two citation sequences with $y_i \leq z_j$ for all i and j, and let $y_{\max}$ denote the largest value in sequence y. Let $x = \langle x_1, \ldots, x_{r+s} \rangle$ denote the union of sequences y and z, and let $p \geq 0$ be an integer.*

(a) If $y_{\max} \leq s+p$ holds, then the $h(p)$-index of sequence x coincides with the $h(p)$-index of sequence z.

(b) If $y_{\max} > s+p$ holds, then the $h(p)$-index of sequence x equals the $h(p+s)$-index of sequence y incremented by value s.

Proof. Suppose $y_{\max} \leq s+p$. Since the $s+1$ largest elements in sequence x are the s elements $z_1, \ldots, z_s$ and $y_{\max}$, in this case the $h(p)$-index of sequence x is at most s. Hence only the elements in sequence z are relevant. This yields (a).

Next suppose $y_{\max} > s + p$. In this case the s elements $z_1, \ldots, z_s$ and element $y_{\max}$ all have value at least $s + p + 1$, and hence the $h(p)$-index of sequence x is at least $s + 1$. Then the $h(p)$-index of x is determined by all s elements in z together with a subset of t elements in y that all have value at least $s + t + p$. Hence we are looking for the largest t such that y contains t elements that all have value at least $s + t + p$; this largest t is precisely the $h(p + s)$-index of y. □

Lemma 2 suggests the following recursive approach for computing the $h(p)$-index of a given sequence $x = \langle x_1, \ldots, x_n \rangle$.

> Phase 1: If $n \leq 2$ then determine the $h(p)$-index directly and stop.
> Otherwise set $s := \lfloor n/2 \rfloor$, and determine the value v of the s-largest element in sequence x.
> Split sequence x into a sequence z of length s that contains elements of value $\geq v$, and into a sequence y of length $n - s$ that contains elements of value $\leq v$.
>
> Phase 2: If $v \leq s + p$ holds, then throw away sequence y. Recursively compute and then output the $h(p)$-index of sequence z.
> If $v > s + p$ holds, then throw away sequence z. Recursively compute the $h(p + s)$-index of sequence y, increment it by s, and output the resulting value.

In the beginning sequence x is stored in sequential memory. Proposition 1 allows us to find the s-largest element in $O(n)$ time. The sequences y and z are easily determined and stored in sequential memory in $O(n)$ time (we run through the tape containing x, and split its contents appropriately into two other tapes; afterwards we may reuse the tape that contained sequence x). In Phase 2 we recurse on a sequence of length at most $\lceil n/2 \rceil$. Since every search step removes half of the search interval, the time complexity $T(n)$ of this procedure satisfies

$$T(n) \leq T(\lceil n/2 \rceil) + O(n).$$

A straightforward induction yields $T(n) \leq cn$ for some appropriate constant c. Finally we note that the h-index coincides with the $h(0)$-index.

Theorem 5. *The h-index of an unordered citation sequence $\langle x_1, \ldots, x_n \rangle$ in sequential memory can be determined in $O(n)$ time. Under this model of computation, no algorithm can have a worst case time complexity that is asymptotically better than $O(n)$.*

Proof. The $O(n)$ time algorithm in the positive part follows from the above discussion. The negative result follows along the lines of the negative result in Theorem 4. □

Now let us turn to the g-index. Similarly as for the h-index, we introduce a purely technical generalization that is defined around two non-negative integer

parameters $p, q \geq 0$: The $g(p,q)$-index of a citation sequence $\langle x_1, \ldots, x_n \rangle$ is the largest integer g such that the sequence contains g elements that have sum at least $(g+p)^2 - q$. Note that the classical g-index coincides with the $g(0,0)$-index.

Lemma 3. *Let $y = \langle y_1, \ldots, y_r \rangle$ and $z = \langle z_1, \ldots, z_s \rangle$ be two citation sequences with $y_i \leq z_j$ for all i and j, and let Z denote the sum of all elements in sequence z. Let $x = \langle x_1, \ldots, x_{r+s} \rangle$ denote the union of sequences y and z, and let p and q be non-negative integers.*

(a) If $Z \leq (s+p)^2 - q$, then the $g(p,q)$-index of sequence x coincides with the $g(p,q)$-index of sequence z.

(b) If $Z > (s+p)^2 - q$, then the $g(p,q)$-index of sequence x equals the $g(p+s, q+Z)$-index of sequence y incremented by value s.

Proof. Suppose $Z \leq (s+p)^2 - q$. Then the $g(p,q)$-index of sequence x is at most s, and only the elements in sequence z are relevant for it. This yields (a).

Next suppose $Z > (s+p)^2 - q$, in which case the $g(p,q)$-index of x is at least s. Then the $g(p,q)$-index of x is determined by all s elements in z, together with a subset of t elements in y. Let Y' denote the sum of these t elements, and observe that $Z + Y' \geq (s+t+p)^2 - q$ must hold true. These t elements in sequence y whose sum is at least $(s+t+p)^2 - q - Z$ precisely yield the $g(p+s, q+Z)$-index of y. □

Lemma 2 leads to the following recursive approach for computing the $g(p,q)$-index of a given sequence $x = \langle x_1, \ldots, x_n \rangle$.

> Phase 1: If $n \leq 2$ then determine the $g(p,q)$-index directly and stop.
> Otherwise set $s := \lfloor n/2 \rfloor$, and determine the value v of the s-largest element in sequence x.
> Split sequence x into a sequence z of length s that contains elements of value $\geq v$, and into a sequence y of length $n - s$ that contains elements of value $\leq v$.
> Determine the sum Z of all elements in sequence z.
>
> Phase 2: If $Z \leq (s+p)^2 - q$ holds, then throw away sequence y. Recursively compute and then output the $g(p,q)$-index of sequence z.
> If $Z > (s+p)^2 - q$ holds, then throw away sequence z. Recursively compute the $g(p+s, q+Z)$-index of sequence y, increment it by s, and output the result.

Similarly as in the computation of the $h(p)$-index, this algorithm for the $g(p,q)$-index can be implemented to run in $O(n)$ time if the sequence x is stored in sequential memory.

Theorem 6. *The g-index of an unordered citation sequence $\langle x_1, \ldots, x_n \rangle$ in sequential memory can be determined in $O(n)$ time. Under this model of computation, no algorithm can have a worst case time complexity that is asymptotically better than $O(n)$.* □

Finally let us discuss the w-index. It is easy to determine the w-index of an unordered citation sequence $\langle x_1, \ldots, x_n \rangle$ in $O(n \log n)$ time: First sort the elements in $O(n \log n)$ time (a classical sorting algorithm like MergeSort will do this also for data in sequential memory). Then apply the $O(n)$ algorithm from Theorem 3. Interestingly this is already the best asymptotic time complexity one can reach in sequential memory.

Theorem 7. *The w-index of an unordered citation sequence $\langle x_1, \ldots, x_n \rangle$ in sequential memory can be determined in $O(n \log n)$ time. Under this model of computation, no algorithm can have a worst case time complexity that is asymptotically better than $O(n \log n)$.*

Proof. The proof of the negative statement is based on an auxiliary problem called PERMUTATION-RECOGNITION: Given n integers $u_1, \ldots, u_n$, decide whether these integers form a permutation of the numbers $1, 2, \ldots, n$. It is known that this problem cannot be solved with a worst case time complexity better than $O(n \log n)$, if the data is stored in sequential memory; see for instance Ben-Or ([2]).

Consider an arbitrary instance $u_1, \ldots, u_n$ of PERMUTATION-RECOGNITION. In a first step compute the sum U of all elements in this instance in linear time $O(n)$. If $U \neq \frac{1}{2}n(n+1)$, we stop right away with the answer NO. Otherwise we move on, and feed the sequence $u_1, \ldots, u_n$ into an algorithm A for computing the w-index. If algorithm A finds that the w-index is at most $n-1$, we stop with answer NO. If algorithm A finds that the w-index is n, we stop with the answer YES.

If there was an algorithm A for the w-index with worst case time complexity better than $O(n \log n)$, this approach would yield an algorithm for PERMUTATION-RECOGNITION with worst case time complexity better than $O(n \log n)$. □

6 Conclusion

In this paper we have discussed the algorithmic complexity of computing the h-index, the g-index, and the w-index in various (standard) models of computation. Our results suggest that the h-index is computationally the easiest index to compute, that the g-index needs some more effort, and that the w-index is the hardest.

We note that our techniques can easily be adapted to yield similar results for other scientific impact indices. Consider for instance the so-called *Kosmulski-index* of a citation sequence (see Kosmulski, [9]): A scientist has Kosmulski-index k, if k is the largest integer such that at least k of his articles have received at least k^2 citations each. An equivalent definition of the Kosmulski-index of a citation sequence $x = \langle x_1, \ldots, x_n \rangle$ is as follows: Define an auxiliary citation sequence $y = \langle y_1, \ldots, y_n \rangle$ by setting $y_i = \lfloor \sqrt{x_i} \rfloor$ for $1 \leq i \leq n$. Then the Kosmulski-index of sequence x coincides with the h-index of sequence y. Now our results on the h-index imply that

- the Kosmulski-index of a sorted citation sequence $x_1 \geq x_2 \geq \cdots \geq x_n$ can be determined in $O(\log_2 n)$ time,

- the Kosmulski-index of an unordered citation sequence in random access memory can be determined in $O(n)$ time,
- the Kosmulski-index of an unordered citation sequence in sequential memory can be determined in $O(n)$ time.

Furthermore, these time complexities are best possible in the respective models of computation. This once again confirms our intuition that the Kosmulski-index is very closely related to the h-index, and that these two indices behave in more or less the same way.

References

[1] Aho, A. V., Hopcroft, J. E. and Ullman J. D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] Ben-Or M., Lower bounds for algebraic computation trees. In *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing (STOC'1983)*, pages 80–86, 1983 .

[3] Blum, M., Floyd, R. V., Pratt, V. R., Rivest, R. L. and Tarjan, R. E. Time bounds for selection. *Journal of Computer and System Sciences* 7:448–461, 1973.

[4] Cormen, T. H., Leiserson, C. E. and Rivest R. L. *Introduction to Algorithms.* MIT Press, 1990.

[5] Egghe, L. An improvement of the h-index: The g-index. *ISSI Newsletter* 2:8–9, 2006.

[6] Egghe, L. Theory and practice of the g-index. *Scientometrics* 69:131–152, 2006.

[7] Gross, P. L. K. and Gross, E. M. College libraries and chemical education. *Science* 66(1713):385–389, 1927.

[8] Hirsch, J. E. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences* 102(46):16569–16572, 2005.

[9] Kosmulski M. A new Hirsch-type index saves time and works equally well as the original h-index. *ISSI Newsletter* 2:4–6, 2006.

[10] Papadimitriou C. H. *Computational Complexity.* Addison-Wesley, 1994.

[11] Woeginger G. J. An axiomatic characterization of the Hirsch-index. *Mathematical Social Sciences* 56:224–232, 2008.

[12] Woeginger G. J. (2008b). An axiomatic analysis of Egghe's g-index. *Journal of Informetrics* 2:364–368, 2008.

Received 6th November 2009

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János